Treball de Fi de Màster

Master's degree in Automatic Control and Robotics

Extension of the BRIEF descriptor for color images and its evaluation in robotic applications

MEMÒRIA

Autor: Àngel Matilla Alastruey
Director: Joan Aranda López
Convocatòria: 01/2020

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Abstract

During the development of the project, new extensions of the BRIEF descriptor are defined based in the use of color information in different color spaces. They are evaluated against each other and the original BRIEF descriptor (that uses intensity pixel values in grayscale images) by means of a matching test of different points of two different images of the same scene. The selected extension is used to recognize objects in a real time robotic application by means of a classification method.

Therefore, the target is to improve the BRIEF descriptor to be applicable in color images by defining the different color extensions and making a comparative evaluation to ensure selecting the best one that most improve the basic BRIEF descriptor with the use of color information. The speed and the recognition rate are computed for every extension and the basic BRIEF to compare the performance of each descriptor.

Ones an extension is selected, it is used the Bag Of Features model to create an algorithm able to recognize objects. The BOF model detects the keypoints with the FAST algorithm, describes them by means of the selected extension of the BRIEF descriptor and classifies them using a linear Support Vector Machine.

Finally, the algorithm is tested in different situations and conditions, varying illumination, a little bit rotation and background, and checking performance with occluded objects.

Index

1. Glossary

BRIEF -> Binary Robust Elementary Features

FAST -> Features from Accelerated Segment Test

BOF -> Bag Of Features

BOVW -> Bag Of Visual Words

SSE -> Sum of Squared Errors

SVM -> Support Vector Machine

WT -> Wiener Transform

2. Preface

2.1. Project origin

The project is based in the original paper of BRIEF descriptor [1], which has been analyzed in detail to understand all the concepts. Also, the same database has been used in some experimental parts.

There is also another paper where the BRIEF descriptor is adapted to use color information of RGB color space [2]. The paper has been found after proposing the two RGB variants of the BRIEF descriptor and it has been very useful to understand how to use the color information from R, G and B channels.

2.2. Motivation

The main motivation is to apply the BRIEF descriptor to color images, by using the different color channels to extract information and verify if it improves the descriptor.

The idea to carry on this project comes from the last lessons of the subject “Advanced Topics in Computer Vision”, where the BRIEF descriptor was introduced. At first, this descriptor seems to be magic as it is easy to compute and it operates better with random distributions of the selected pixels.

Furthermore, ESAII department of UPC was developing a project of an automated kitchen, and it was a big motivation to develop a descriptor to recognize kitchen objects in an efficient way.

2.3. Previews requirements

The necessary requirements are the knowledge and comprehension of local descriptors for Computer Vision but also, an important requirement is to have a good base in Machine Learning knowledge.

3. Introduction

BRIEF descriptor is a binary grayscale descriptor used in many computer vision applications. As it does not consider color information, it can be a big space to make research by improving this descriptor.

Basically, it makes a quantity of intensity values difference between two pixels of a grayscale image.

3.1. Project aim

The main target of the project is to improve the BRIEF descriptor, defining variants of the descriptor by using color information to get richer descriptors. The idea is to use two color spaces: RGB and YCbCr. For each color space, two descriptors are defined, so four descriptors are created:

- RGB BRIEF descriptor: each intensity value difference is done in the same RGB channel.
- Mixed RGB BRIEF descriptor: the differences can be done with pixel intensity values of different RGB channels.
- YCbCr BRIEF descriptor: each intensity value difference isdone in the same YCbCr channel.
- Mixed YCbCr BRIEF descriptor: the differences can be done with pixel intensity values of different YCbCr channels.

The extensions (detailed in chapter 8) and the original BRIEF descriptors are used to recognize objects in a Computer Vision application to evaluate their performance. By looking for the speed and the recognition rate, all descriptors are compared to analyze the improvements of the new descriptors and the advantages and drawbacks of each one for a popular robotic application usecase: object recognition.

The project is based in the original paper of the BRIEF descriptor [1] but also in other scientific papers where different authors have been doing research in color extensions of BRIEF descriptor.

3.2. Scope of the project

First, the procedure of the original BRIEF descriptor is explained in detail to be able to understand later the color extensions defined. Before defining the color variants of the descriptor, the principles of the RGB and YCbCr color spaces are also described to understand the composition of each channel and how these color spaces forms the color images.

As one of the main parts of the project, the four color BRIEF extensions are defined to create new descriptors applicable in color images. Then, the four extensions and the original descriptor are compared in a matching application, where two images of the same scene (rotation variance) are compared by extracting first keypoints and after that, a matching operation is done to pair the points that are the same. The performance of the different compared descriptors is measured by means of two behaviour skills: recognition rate and speed.

A real application of object recognition is defined and carried out to apply the selected descriptor in a practical way.

4. BRIEF descriptor

All descriptor variants of this project are based in the BRIEF descriptor. This chapter aims to explain the methodology of the BRIEF descriptor by means of the original paper of the BRIEF descriptor [1].

BRIEF is the acronym of Binary Robust Independent Elementary Features and is an efficient feature point descriptor that returns a binary string, which is computed by simple intensity difference tests. This type of descriptor is being increasingly used in a lot of Computer Vision applications such as object recognition (which is the application tested in the project), 3D reconstruction, camera localization… The big advantage of BRIEF descriptor is that it is a binary descriptor and for that reason, the descriptor building and matching are very fast. For the part of matching, it uses the Hamming distance instead of Euclidean distance that it is very efficient to compute by simply using a XOR logical operation followed by a bit count. The descriptor is calculated for a small patch that surrounds the keypoint.

For calculating the descriptor itself: given a patch p of size S x S centered on the keypoint, a test τ of two pixel intensities on x y positions relative to the center of the patch is the result of:

$$\tau(\mathbf{p};\mathbf{x},\mathbf{y}) := \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

(Eq. 4.1)

Where p(x) is the pixel intensity of a smoothed version of p at x position (x are the coordinates of first point and y the coordinates of the second) relative to the keypoint. Choosing a set of n_d (x y)-locations pairs defines a set of binary tests that will form the descriptor. The BRIEF descriptor will be a n_d-dimensional bitstring:

$$f_{n_d}(\mathbf{p}) := \sum_{1 \le i \le n_d} 2^{i-1}\, \tau(\mathbf{p};\mathbf{x}_i,\mathbf{y}_i)$$

(Eq. 4.2)

The length of the descriptor n_d can be parameterized to have good compromises between speed, storage efficiency and recognition rate. The paper concludes that for matching purposes, 256 comparisons are necessary to equal the recognition rate of other descriptors such as SURF or SIFT but 128 are sufficient for some cases too. It is decided to use 128 comparisons in the project as it is faster and gives good results.

By construction, the pixel tests consider only the information at single pixels and are therefore very noise-sensitive. By pre-smoothing the patch with a Gaussian filter, this sensitivity can be reduced, thus increasing the stability and repeatability of the descriptors. In the original paper it is checked different values of variances of the Gaussian kernel and it concludes that between 1 and 3 is quite constant and at 2 the recognition rate performance starts to saturate, so in practice it decides to use a value of 2 for variance.

The spatial arrangement of the binary tests in the S x S patches is also tested in the original paper to determine the best distribution. The origin of the patch is considered in the center which is the location of the keypoint. An experiment with 5 different distributions is carried on and the comparison is done by calculating the recognition rate.

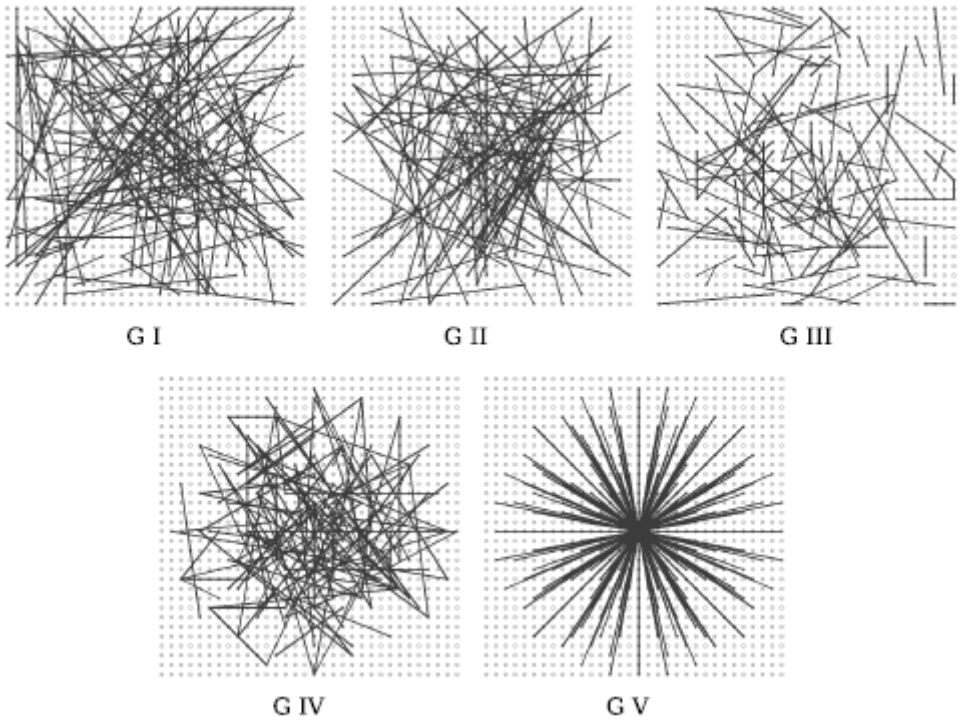

Fig. 4.1. Spatial arrangement of the binary test in S x S patch. [1]

I. Uniform distribution: the x y locations are evently distributed over the patch window, and test can appear anywhere, even near borders.

II. Gaussian distribution: tests are sampled from an isotropic Gaussian distribution.

III. Two Gaussian distributions: The first location x is sampled from a Gaussian centered around the origin while the second location is sampled from another Gaussian centered on x. This method forces the test to be local. The tests locations out of the patch are forced to the edge of the patch.

IV. The x y locations are randomly sampled from discrete locations of a coarse polar grid introducing a spatial quantization.

V. All x are taken in the center of the patch (0,0) and y takes all possible values on a coarse polar grid containing n_d points.

The one that gives the worst results by far is the G V which is the only one that has a symmetrical and regular strategy. All the others give similar recognition rate results with G II (Gaussian distribution) enjoying a small advantage, which confirms that random distributions behave clearly better.

The tests of original paper are done with a public database of six test image sequences for evaluating robustness to: view point changes (wall, graffiti and fountain), compression artifacts (Jpeg), illumination changes (Light) and image blur (Trees). Each sequence contains five image pairs, with the first image being the same in all pairs and the second image shot from a monotonically growing baseline, which makes matching increasingly more difficult. For evaluating and comparison purposes, two metrics are used: elapsed CPU time and recognition rate. The first one is the average time over many repeated runs, and the recognition rate, given an image pair, is computed as:

- Pick N interest points from first image, infer the N points in the other image of the pair from the ground truth data, and compute the 2N associated descriptors using the method under consideration.

- For each point in the first set, find the nearest neighbor in the second one and call it a match.

- Count the number of correct matches n_c and compute the recognition rate as $r=n_c/N$.

5. RGB color space

A color space is a color interpretation system, a specific organization of the colors in an image or video. The RGB color space use color primaries defined by the additive RGB color model. This model uses the addition of red, blue and green light in different ways to reproduce a broad array of colors. The name of the model comes from the initials of the three primary colors, red, green and blue. The main purpose of the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography. Before the electronic age, the RGB color model already had a solid theory behind it, based in human perception of colors.

The combination of these three primary colors (R, G and B channels) can not produce all colors; it can produce only colors within the color 2D triangle defined by the chromacities of the primaries which are reproduced by adding non-negative amounts of those colors of light.

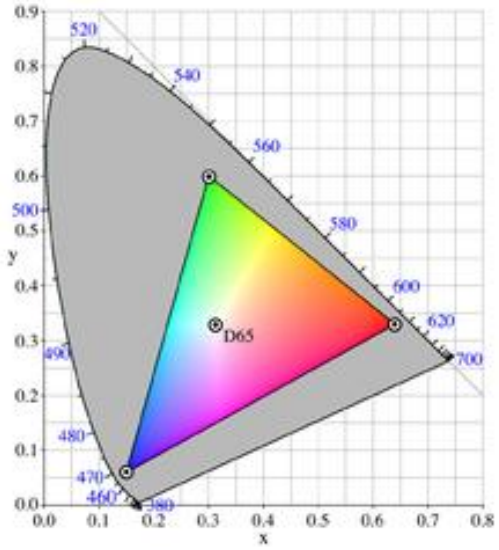

Fig. 5.2. 2D triangle defined by the chromacities of the primary

Another representation of the possibilities for mixing the three primary colors can be represented as a three dimensional coordinate plane with the values for red, green and blue on each axis. This range of values is called the color depth and is the quantization of the possible values per component, that in the case of RGB color space it is done using integer

numbers from 0 to some power of two minus one (2n − 1) to fit them into some bit groupings. This coordinate plane yields a cube called the RGB color space:

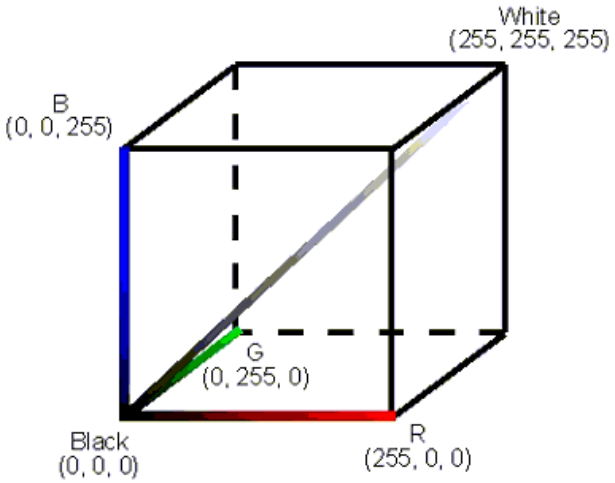

Fig. 5.2. Cube representation of the RGB color space.

The RGB color model is implemented in different ways, depending on the capabilities of the system used. By far the most common is the 24-bit implementation, with 8 bits, or 256 discrete levels of color per channel. Any color space based on such a 24-bit RGB model is thus limited to a range of 256×256×256 ≈ 16.7 million colors.

6. YCbCr color space

The YCbCr is another color space, another interpretation of the color system to represent the colors in a different way. The difference between YCbCr and RGB is that YCbCr represents color as brightness and two color difference signals, while RGB represents color as red, green and blue. In YCbCr, the Y is the brightness (luma), Cb is blue minus luma (B-Y) and Cr is red minus luma (R-Y).

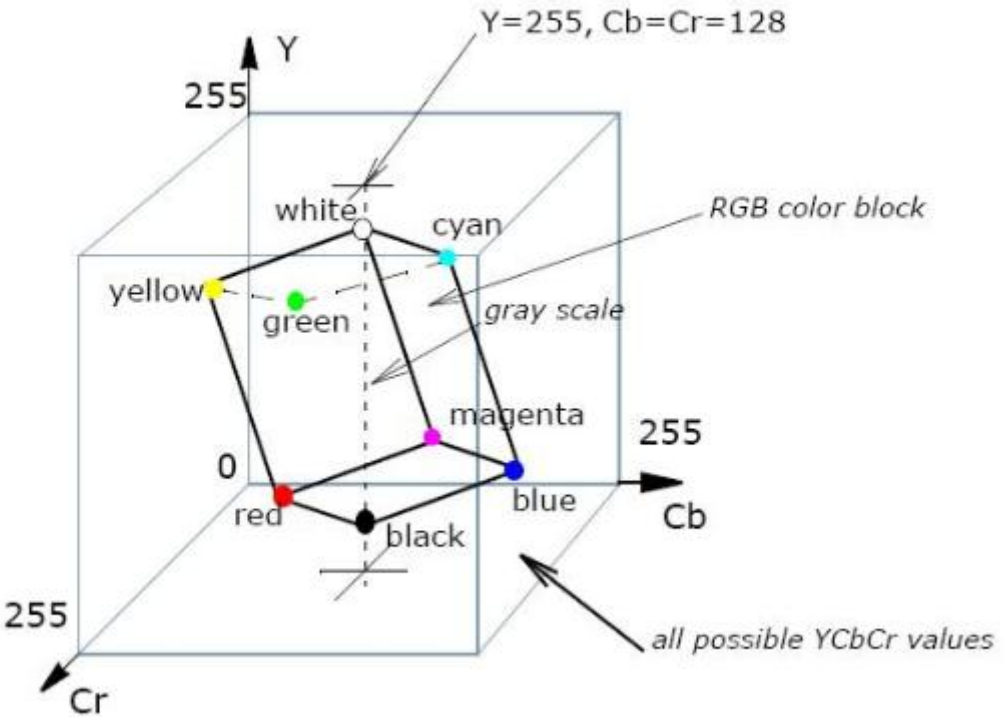

Fig. 6.1. 3D representation of the YCbCr color space.

YCbCr is used to separate out a luma signal (Y) that can be stored with high resolution or transmitted at high bandwidth, and two chroma components (Cb and Cr) that can be bandwidth-reduced, subsampled, compressed, or otherwise treated separately for improved system efficiency. One practical example would be decreasing the bandwidth or resolution allocated to "color" compared to "black and white", since humans are more sensitive to the black-and-white information (see image example bellow). This is called chroma subsampling:

Fig. 6.2. Up-left original picture, up-right Y channel, down-left Cb channel and down-right Cr channel.

The principal advantage of the YCbCr model in image processing is decoupling of luminance and color information. The importance of this decoupling is that the luminance component of an image can be processed without affecting its color component. If the luma component Y' is fixed in a constant value, the Cb and Cr components can be represented as a plane:

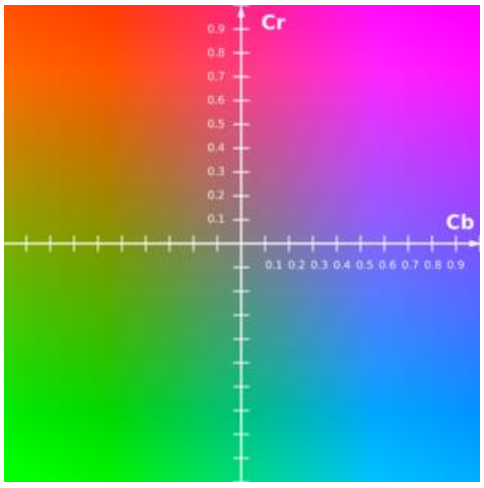

Fig. 6.3. The CbCr plane at constant luma Y'=0.5.

Digital Y′CbCr (8 bits per sample) is derived from analog R'G'B' as follows:

$$\begin{aligned} Y' &= 16 + (65.481 \cdot R' + 128.553 \cdot G' + 24.966 \cdot B') \\ C_B &= 128 + (-37.797 \cdot R' - 74.203 \cdot G' + 112.0 \cdot B') \\ C_R &= 128 + (112.0 \cdot R' - 93.786 \cdot G' - 18.214 \cdot B') \end{aligned} \quad \text{(Eq. 6.2)}$$

Here, the prime (′) symbols mean gamma correction is being used; thus R′, G′ and B′ nominally range from 0 to 1, with 0 representing the minimum intensity and 1 the maximum.

7. Color extensions of the BRIEF descriptor

Entering in the most important part of the project, four different color extensions of the BRIEF descriptor are defined in the current chapter. In these extensions, the color information is used instead of using only grayscale images, to improve the performance of the BRIEF descriptor. For building the extensions, the RGB and YCbCr color spaces are used.

The original BRIEF descriptor implementation is obtained from reference [3], and their extensions are developed based in this implementation.

7.1. RGB BRIEF descriptor

In this extension, it is used the RGB color space for the color information of the BRIEF descriptor extension. The color information is introduced when the BRIEF descriptor computes the pixel intensity differences. In the case of the original BRIEF descriptor, the pixel intensity is obtained from the grayscale image. This extension has already been defined in reference [2], where it is checked that improves the original BRIEF descriptor.

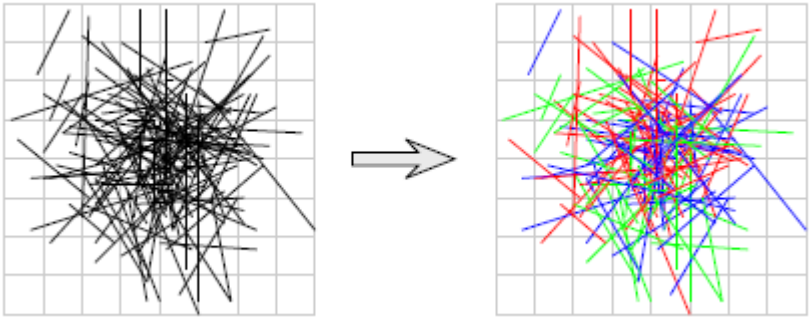

Fig. 7.1. Left BRIEF descriptor, right RGB BRIEF descriptor. [2]

Now in the RGB BRIEF descriptor, the image is not converted to grayscale but it is manipulated in RGB color image format. So there are three channels of color information: red, green and blue. The pixel intensity differences are computed in one of the three channels, maintaining the channel in the two pixels of the difference. The channel is selected randomly with the same random distribution than for the spatial distribution of the subtracted pixels in the descriptor window around the keypoint. The mathematical formulation is:

$$(\mathbf{x}_i, \mathbf{y}_i) \rightarrow (\mathbf{x}_i, \mathbf{y}_i, \mathbf{c}_i)$$

(Eq. 7.3)

$$\tau(\mathbf{p}, \mathbf{x}, \mathbf{y}, \mathbf{c}) = \begin{cases} 1 & \text{if } \mathbf{p_c}(\mathbf{x}) < \mathbf{p_c}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

(Eq. 7.2)

Here, $\mathbf{x_i}$ are the coordinates of the first point and $\mathbf{y_i}$ the coordinates of the second point of the binary tests. The channel $\mathbf{c_i}$ Є {R;G;B}, and it is selected randomly when generating the pairs. $\mathbf{p_c}$ is the intensity of a pixel at a given location relative to the center of the patch in a given color channel. The selection of $\mathbf{c_i}$ is fixed and stored together with $\mathbf{x_i}$ and $\mathbf{y_i}$.

The descriptor has the following inputs and outputs for its computation:

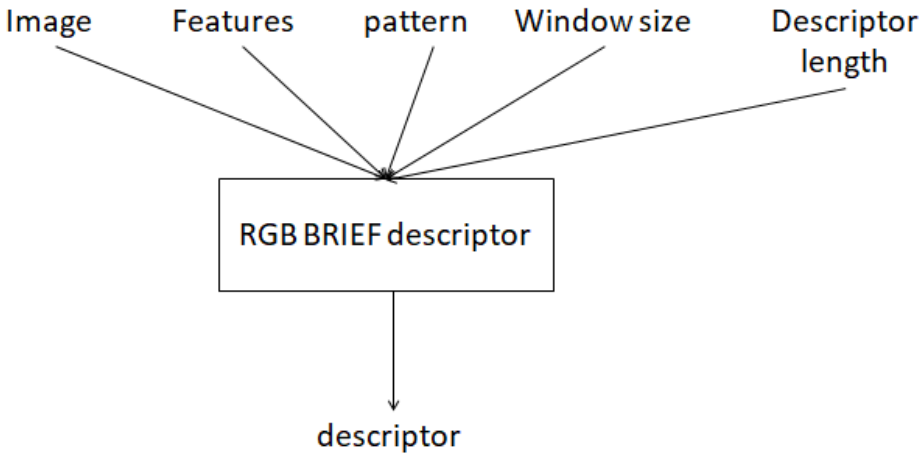

Fig. 7.2. Inputs and outputs of RGB BRIEF descriptor.

The inputs are the analyzed image, the feature points location (m x 2 matrix of row column coordinates), the random pattern of pixels location (for the binary tests) and channel selection, the window size around the keypoint and the BRIEF descriptor length. The pattern input is an n x 5 matrix of points to be compared around the feature point with 1 and 2 columns as the coordinates of the 1st point (x point = row and column) and the 3 and 4 columns as the coordinates of the 2nd point (y point = row column), with random channel in 5 column.

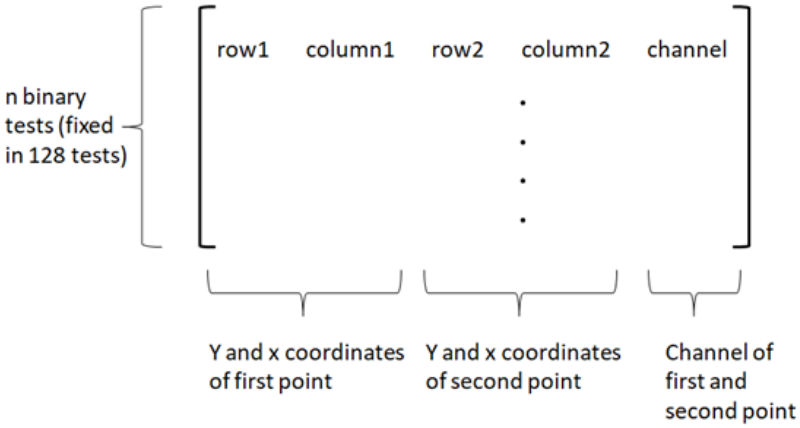

Fig. 7.3. Random pattern matrix that contains pixel locations and channel selection.

The output is a matrix of m x n, where m are all the descriptors of the different feature points and n is the descriptor length, so that each row describes the corresponding feature point.

In this way, the descriptor vector is composed of pixel intensity differences from different RGB channels selected randomly and maintaining the same channel for the two pixels of a difference. The descriptor length is the same than the other extensions and the original BRIEF, so it will have color information of each pixel intensity difference but only of one channel per difference.

7.2. Mixed RGB BRIEF descriptor

The mixed RGB BRIEF descriptor is a BRIEF descriptor extension that also uses the RGB color space, and the concept of mixed is used to specify that the channel is not forced to be the same for the two pixel of a binary test. Another time, the spatial arrangement of the test pixels and the selection of the channels are chosen randomly with a distribution according to the results of original paper in chapter 4 and the tests of chapter 8.

The inputs and outputs for the descriptor computation are the same than the previous extension, but the pattern input is an n x 6 matrix of points to be compared around the feature point with 1 and 2 columns as the coordinates of the 1st point (point x = row and column) and the 3 and 4 columns as the coordinates of the 2nd point (point y = row and column), with first point random channel in 5 column and second point random channel in 6 column.

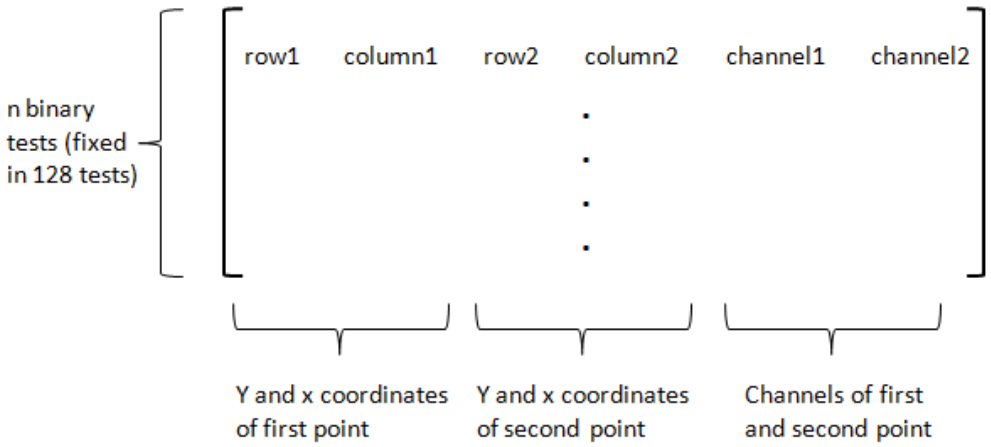

Fig. 7.4. Random pattern matrix that contains pixel locations and channel selection.

The mathematical formulation of the Mixed RGB BRIEF descriptor is the following one:

$$\left(x_i, y_i\right) \rightarrow \left(x_i, y_i, c_{x_i}, c_{y_i}\right) \qquad \text{(Eq. 7.3)}$$

$$\tau\left(p, x, y, c_x, c_y\right) = \begin{cases} 1 \text{ if } p_{c_x}(x) < p_{c_y}(y) \\ 0 \quad \text{otherwise} \end{cases} \qquad \text{(Eq. 7.4)}$$

Where $\mathbf{c_{xi}}$ and $\mathbf{c_{yi}}$ Є {R;G;B} are the channels of point $\mathbf{x_i}$ and $\mathbf{y_i}$, and they are selected randomly separately. $\mathbf{p_{cx}}$ and $\mathbf{p_{cy}}$ are the intensity of a pixel at given locations relative to the center of the patch in a given color channel. The selection of $\mathbf{c_{xi}}$ and $\mathbf{c_{yi}}$ is fixed and stored together with $\mathbf{x_i}$ and $\mathbf{y_i}$.

7.3. YCbCr BRIEF descriptor

Now, it is changed the color space to the YCbCr color space. This change is done for deleting the luma component Y and making in this way, the descriptor invariant to illumination. The channel is maintained for x and y points and selected randomly as in the previous extensions. With experiment of chapter 8 it is tested with only channels Cb and Cr and the recognition rate results are bad, it was expected to outperform previous extensions but the performance is very similar to the original BRIEF descriptor. So for that reason, it is decided to include the luma component Y in the extension descriptor:

Descriptors	Recognition rate
Original BRIEF	0.4694
YCbCr (without Y)	0.5556
YCbCr	0.8302

Table. 7.1. Recognition rate results of experiment from chapter 8 for different variants of YCbCr BRIEF descriptor

The above results come from conditions and setup of experiment in chapter 8.

In this extension, the channel Y, Cb or Cr is selected randomly as in previous extensions, and is the same channel for both points (x and y) of a binary test.

The inputs and outputs for the descriptor computation are the same than the previous extension RGB BRIEF descriptor, with the pattern input being an n x 5 matrix of points to be compared around the feature point with 1 and 2 columns as the coordinates of the 1st point (point x = row and column) and the 3 and 4 columns as the coordinates of the 2nd point (point y = row and column), with random channel in 5.

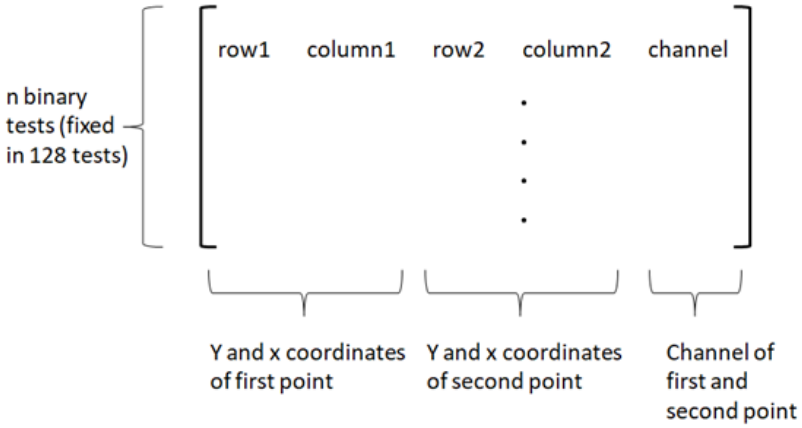

Fig. 7.5. Random pattern matrix that contains pixel locations and channel selection.

The mathematical formulation of the YCbCr BRIEF descriptor is the following one:

$$(\mathbf{x}_i, \mathbf{y}_i) \rightarrow (\mathbf{x}_i, \mathbf{y}_i, \mathbf{c}_i) \tag{Eq. 7.5}$$

$$\tau(\mathbf{p}, \mathbf{x}, \mathbf{y}, \mathbf{c}) = \begin{cases} 1 & \text{if } \mathbf{p_c}(\mathbf{x}) < \mathbf{p_c}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases} \tag{Eq. 7.6}$$

Here, $\mathbf{x_i}$ are the coordinates of the first point and $\mathbf{y_i}$ the coordinates of the second point of the binary tests. The channel $\mathbf{c_i}$ Є {Y;Cb;Cr}, and it is selected randomly when generating the pairs. $\mathbf{p_c}$ is the intensity of a pixel at a given location relative to the center of the patch in a given color channel. The selection of $\mathbf{c_i}$ is fixed and stored together with $\mathbf{x_i}$ and $\mathbf{y_i}$.

7.4. Mixed YCbCr BRIEF descriptor

This final extension is also computed in YCbCr color space. Now, it is decided to continue without the luma Y channel to have one descriptor with only two channels, as all the others have three (R, G and B or Y, Cb and Cr) and the original BRIEF one channel, the grayscale picture.

As its name says, the channel can be mixed between the two points of one binary test, that is, the channel does not have to be the same for the points x and y of a binary test. Another time, the spatial arrangement of the test pixels and the selection of the channels are chosen randomly with a distribution according to the results of original paper in chapter 4 and the tests of chapter 9.

The inputs and outputs for computing the descriptor are the same than the previous extension, but the pattern input is an n x 6 matrix of points to be compared around the feature point with 1 and 2 columns as the coordinates of the 1st point (point x = row and column) and the 3 and 4 columns as the coordinates of the 2nd point (point y = row and column), with first point random channel in 5 column and second point random channel in 6 column.

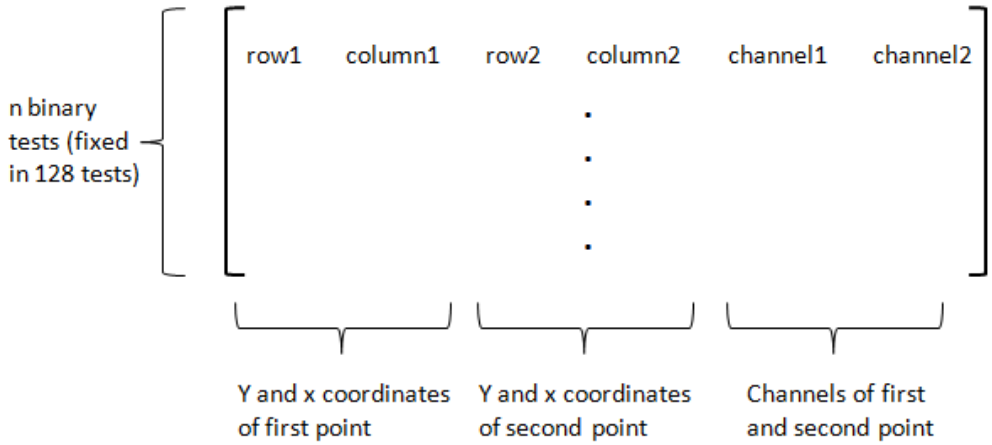

Fig. 7.6. Random pattern matrix that contains pixel locations and channel selection.

The mathematical formulation of the Mixed YCbCr BRIEF descriptor is the following one:

$$\left(x_i, y_i\right) \rightarrow \left(x_i, y_i, c_{x_i}, c_{y_i}\right) \quad \text{(Eq. 7.7)}$$

$$\tau\left(p, x, y, c_x, c_y\right) = \begin{cases} 1 \text{ if } p_{c_x}(x) < p_{c_y}(y) \\ 0 \quad \text{otherwise} \end{cases} \quad \text{(Eq. 7.8)}$$

Where $\mathbf{c_{xi}}$ and $\mathbf{c_{yi}}$ Є {Cb;Cr} are the channels of point $\mathbf{x_i}$ and $\mathbf{y_i}$, and they are selected randomly separately. $\mathbf{p_{cx}}$ and $\mathbf{p_{cy}}$ are the intensity of a pixel at given locations relative to the center of the patch in a given color channel. The selection of $\mathbf{c_{xi}}$ and $\mathbf{c_{yi}}$ is fixed and stored together with $\mathbf{x_i}$ and $\mathbf{y_i}$.

8. Selection methodology of the best extension

An experiment is carried out to compare and evaluate each of the extensions of the BRIEF descriptors and the original one. The clue is to apply all the descriptors in the same conditions and obtain some results to be able to compare them and finally select the “best” one. In this section, the experiment is described in detail.

8.1. Experiment database

Each extension is compared against each other and the original BRIEF by means of a picture matching experiment. For the experiment it is used the same database of the original BRIEF paper, in particular, the Graffiti database:

Fig. 8.3. First two images of the Graffiti database taken from reference [6].

As it can be seen in figure 8.1, the database consists in a photo of a graffiti in a wall and then, four more photos in different points of view are taken by rotating the camera.

8.2. Evaluation methodology

For the evaluation part, two metrics are computed as in original BRIEF paper: elapsed CPU and recognition rate. For the first one, the delayed time for computing the descriptors for all feature points of the two images is measured five times (after the three first executions) and an average is done. For the recognition rate metric, the following steps have been done as in the original paper:

- Feature detection: Pick N interest points from the first image.
- Infer the N corresponding points in the other from the ground truth data (homography matrix).
- Keypoints description: compute the 2N associated descriptors using the method under consideration.
- Features matching: for each point in the first set, find the nearest neighbor in the second one and call it a match.
- Count the number of correct matches n_c and take the recognition rate to be $r=n_c/N$.

The main program to develop the matching tests is described below:

First, to obtain the interest points, it is used the FAST feature detector which is one of the fastest and it is recommended in the original paper. To reduce and fix the number of keypoints, it is set to save the 400 strongest points. Also, to avoid having a lot of nearby points, it is established a condition that deletes points with an Euclidean distance smaller than a fixed reference and keeps only the first one. In this manner, it is ensured that the distribution of the points in the image is uniform.

Ones we have the features of the first image, it is used the homography matrix provided in the database to project the points in their corresponding location in the second image. This matrix has rotation, translation and scale information about the transformation between two images of the same scene. The points projected near the borders or outside the second image are discarded in both images.

```
 8.7976964e-01   3.1245438e-01  -3.9430589e+01
-1.8389418e-01   9.3847198e-01   1.5315784e+02
 1.9641425e-04  -1.6015275e-05   1.0000000e+00
```

Fig. 8.2. Homography matrix for the first two images of the Graffiti database.

Now, the BRIEF descriptors for the two sets of points are computed. The following input parameters are defined based in the original BRIEF paper conclusions (chapter 4): the descriptor length (fixed in 128) and the variance of the Gaussian filter used inside the descriptor function which is fixed with a value of 2. The window size around the feature point is defined as 11 x 11 pixels window, and the random pattern that establishes the pixels to be compared and the channel of the color space (for each pixel or the same channel for both, depending on the BRIEF extension) is computed and defined according to original BRIEF

paper conclusions. This random pattern is generated with rng(seed, generator) Matlab function that can fix the generated random number to be able to use the same distances in all codes, this is very important because the pixels location of the binary tests have to be the same for all keypoints descriptions. Defining rng(0,'v5normal'), the generator is a Gaussian function with 0 mean, which is the specification of the original BRIEF paper.

For the matching of the points of the two images, it is used the code of the original BRIEF implementation, but it is added the left-right consistency check: it consists on looking for the nearest neighbor of each point of the first image among the points of the second one, and also in reverse. Then, only the pairs that agree in the two searches are considered as a match.

To count the number of good matches, the index of each point of the pair match must be equal, as it is known that the list of features of the first image and their projection in the second image are in the same order. For example good matches are [1 1] or [32 32].

8.3. Results

With the conditions described above, it is developed the first comparison experiment: by using the final FAST points location, the image where the FAST points are obtained, the normal distribution for random numbers, a window size of 11 x 11 and a descriptor length of 128. For computing elapsed CPU, the time needed for computing the descriptors of the two sets of FAST points is measured 5 times in seconds:

Original BRIEF: 0.132385, 0.141498, 0.147767, 0.160187, 0.136846.

RGB: 0.248346, 0.244360, 0.241209, 0.239224, 0.243400.

Mixed RGB: 0.242647, 0.243577, 0.267407, 0.242134, 0.245966.

YCbCr: 0.245176, 0.252568, 0.247436, 0.265924, 0.245000.

Mixed YCbCr: 0.217300, 0.216671, 0.234498, 0.206646, 0.211933.

And it is done an average of these times.

Descriptors	Recognition rate	Elapsed CPU (seconds)
Original BRIEF	0.4694	0.143736
RGB	0.6087	0.243307
Mixed RGB	0.6429	0.248346
YCbCr	0.8302	0.251221
Mixed YCbCr	0.4857	0.217409

Table. 8.4. Results of Selection methodology experiment with Gaussian distribution generator for random numbers. In green the best and in red the worst.

Analyzing the results, the order (better to worse) according to the recognition rate is: YCbCr, mixed RGB, RGB, mixed YCbCr and original BRIEF. In terms of speed, the fastest one is the original BRIEF descriptor since it only operates in one channel as it works with grayscale images. The ones that take color information from three channels (RGB, Mixed RGB and YCbCr) are the slowest ones with speeds between 0.243 and 0.251. Finally, the Mixed YCbCr is a little bit faster than the other variants of the original BRIEF because it only uses the channels Cb and Cr.

Now, all different random numbers generators of Matlab software are tested with mean 0 and their recognition rates are compared. The following table shows the recognition rate results using the different Matlab options for the distribution function generators for the random numbers and the average of them. All tests are carried on for all different descriptor variants and the original BRIEF descriptor:

Descriptors	twister	simdTwister	combRecursive	multFibonacci	v5uniform	v5normal	v4	Average
Original BRIEF	0.488	0.4894	0.3778	0.4318	0.4468	0.4694	0.4762	0.4542
RGB	0.6136	0.5714	0.5714	0.5455	0.5745	0.6087	0.561	0.578
Mixed RGB	0.6087	0.7308	0.7818	0.66	0.6818	0.6429	0.5682	0.6677
YCbCr	0.7391	0.7719	0.72	0.8727	0.7869	0.8302	0.7797	**0.7858**
Mixed YCbCr	0.6	0.5641	0.4857	0.5682	0.5778	0.4857	0.5814	0.5518

Table. 8.2. Results of Selection methodology experiment with different generators for random numbers. In green the best and in red the worst.

Observing the above table, it is clear that the best descriptor in terms of recognition rate is still the YCbCr BRIEF descriptor. The average column shows that it has a quite big advantage with respect to the others descriptors.

As a result of this chapter, the best descriptor in terms of recognition rate is the YCbCr BRIEF descriptor with a quite big advantage but it is the slowest one. As expected, the fastest one is the original BRIEF descriptor as it only actuates in one channel (grayscale) but it is not a big difference considering that these times include the blur of the image with a Gaussian filter and the padding of the image as preprocessing steps, and computing a big amount of descriptors of the two matching test images. For that reason, the selected variant is the YCbCr BRIEF descriptor, which will be used in the rest of the project. For this variant and with these test conditions, the best random numbers generator is the Multiplicative Lagged Fibonacci (multFibonacci), this generator outperforms a little the Gaussian one but as in original paper [1] the results of a lot of tests gives advantage to Gaussian, it is decided to use the normal distribution function generator for random numbers.

Now, to see the improvement between original BRIEF descriptor and the YCbCr BRIEF descriptor, it is shown the matching exercise results in the image pair with a Gaussian distribution function generator for random numbers:

Fig. 8.3. Matching results with original BRIEF descriptor.

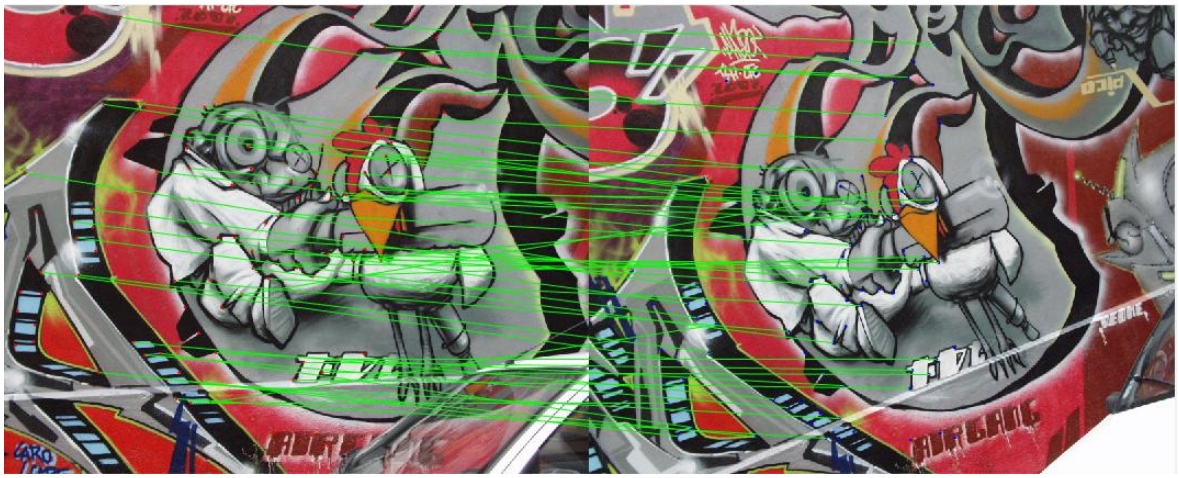

Fig. 8.4. Matching results with YCbCr BRIEF descriptor.

9. Robotic application: Object recognition

The objective of this chapter is to apply the selected YCbCr BRIEF descriptor into a real application of object recognition to evaluate the efficiency of the descriptor in this type of applications. It has been used kitchen objects for applying the developed algorithm in an automated kitchen.

9.1. Bag of Features

For the object recognition task, it is used a Bag Of Features model (BOF) or also called Bag Of Visual Words (BOVW). This model is based on orderless collections of quantized local image descriptors. The origin of the BOF model is found in texture recognition problems. Texture is characterized by the repetition of basic elements or textons:

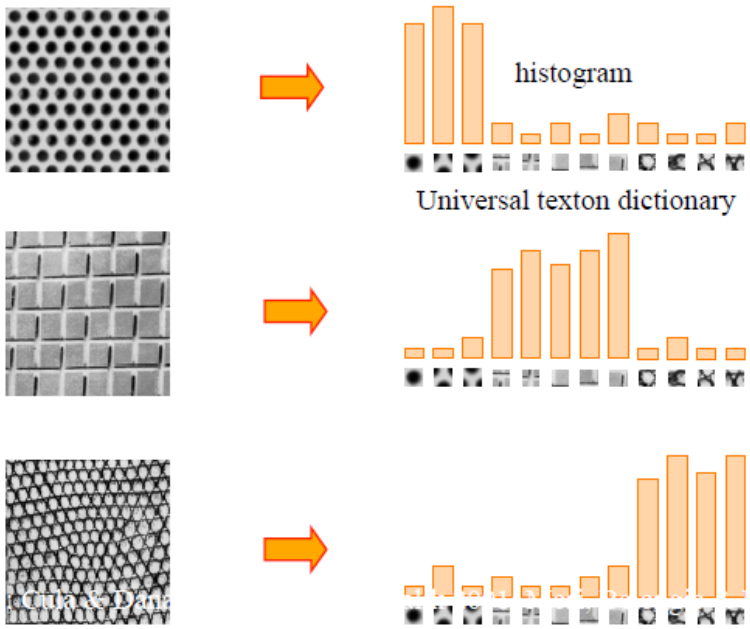

Fig. 9.1. Texton histograms representation.

The origin also can be found in the Bag Of Words model for that reason it is called also Bag Of Visual Words. The orderless document representation is done by counting the occurrence of words within a text. It involves two steps:

- Building of a vocabulary/dictionary of known words.
- A measure of the presence of known words.

This model is used to classify text documents in categories.

A Bag Of Features model represents images as orderless collections of local features, for that orderless of the features it is called a “bag”. A visual vocabulary is constructed to represent the dictionary by clustering features extracted from a set of images. Analyzing different references, this set of images can be one model image per object or different training images of the objects (more than one image per object).

At a high level, the procedure for generating a Bag of Features image representation is summarized as follows:

- **Build Vocabulary**: Extract features from all images in a training set or model images of dataset. There are a lot of feature extractors for keypoint detection as Harris corners, FAST, SIFT, etc. The keypoints are described by means of descriptors such as SIFT, SURF, BRIEF, etc. Vector quantize, or cluster, these features into a “visual vocabulary,” where each cluster represents a “visual word” or “term.” In some works, the vocabulary is called the “visual codebook.” Terms in the vocabulary are the codes in the codebook. The “mean” of each cluster is the visual word.

Fig. 9.2. Features extracted.

Fig. 9.3. Vocabulary of Visual Words.

- **Assign Terms**: Extract features from a novel image. Use Nearest Neighbor or a related strategy to assign the features to the closest terms in the vocabulary.

- **Generate Term Vector**: Record the counts of each term that appears in the image to create a normalized histogram representing a “term vector.” This term vector is the Bag of Features representation of the image. The “term vector” is also called in the project the BOF histogram.

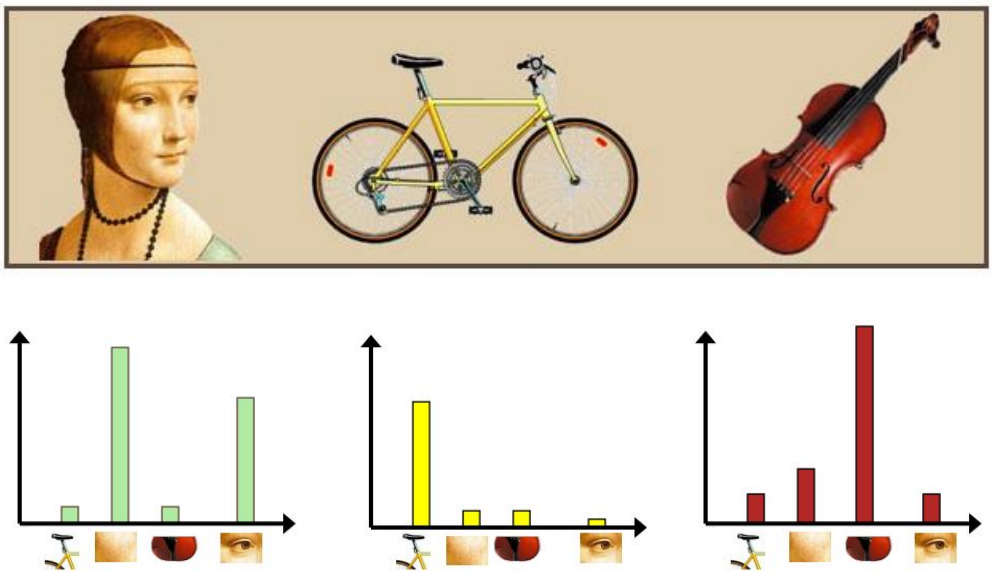

Fig. 9.4. BOF histograms.

Now, for object recognition purposes, these BOF histograms can be used to train a classifier. Then, the classifier can predict the category of a new image by representing first the image with a BOF histogram.

9.2. Clustering methods for categorical data

In the BOF model, as it has been detailed in previous chapter, there is one step where the features are clustered. In this project, as the BRIEF descriptor outputs a binary vector, the data that have to be clustered is categorical.

Categorical variables are characterized by containing a finite number of possible categories (values). In the case of two categories, it is called binary variable which is the case of this project.

Unlike for continuous data, the algorithms for clustering categorical data are scarce. After a

big scouting exercise, it has been found two of the algorithms that cluster categorical data. The equivalent of kmeans for categorical data is the kmodes algorithm [5]. Another method for carrying out the clustering is applying first the Wiener Transform. This last method was obtained from reference [4] and it is applied only for binary data.

9.2.1. Kmodes

The kmodes algorithm [5] is the equivalent kmeans for categorical data. It follows the same procedure but instead of computing the means of each class, it computes modes, and instead of using the Euclidean distance, it uses the hamming distance.

The algorithm follows the next steps:

- Select k initial modes, one for each cluster.
- Compute the hamming distance from each data vector to all cluster modes.
- Assign each data vector to the nearest cluster, which is the one that has the minimum distance.
- Recompute the modes for every cluster.
- Recompute the nearest cluster for all data vectors. If one data vector changes its nearest cluster, recomputed the mode of the previous cluster and the mode of the new cluster.
- Repeat last step until there is no change of cluster.

9.2.1.1. Hamming distance

The distance measure used for categorical data is the Hamming distance. This distance measure counts the number of mismatches of the corresponding feature categories between two categorical vectors. In the case of use, there are only two categories for all features of each feature vector, 0 or 1 as they are binary vectors. The Hamming distance is formulated as:

$$d_1(X, Y) = \sum_{j=1}^{m} \delta(x_j, y_j)$$

(Eq. 9.4)

where

$$\delta(x_j, y_j) = \begin{cases} 0 & (x_j = y_j) \\ 1 & (x_j \neq y_j) \end{cases}$$

(Eq. 9.2)

Where X and Y are two categorical vectors.

9.2.1.2. Mode of a set of categorical vectors

The mode of a set of categorical vectors is a vector composed of the most frequent category of each feature/attribute. For this reason, the mode does not have to be one of the feature vectors of a cluster. It is formulated as the vector Q that minimizes:

$$D(\boldsymbol{X}, Q) = \sum_{i=1}^{n} d_1(X_i, Q)$$

(Eq. 9.3)

Where X is the set of vectors of the cluster.

9.2.1.3. Implementation for binary feature vectors

There is not an implementation of kmodes clustering for Matlab software, so it has been implemented manually. The implemented function considers only categorical data of two categories, hence binary data.

The inputs of the function are a matrix of the feature binary vectors (data) as row vectors, and the number of k clusters (nmodes). The function outputs the index of each binary vector (indcluster) that indicates its cluster, the final modes for each cluster (modes) and the cost of the clustering (cost) which is explained bellow in chapter 9.2.1.4.

Fig. 9.5. Inputs and outputs of kmodes algorithm.

Starting with the implemented code, first k initial random modes are defined. Number of BRIEF descriptors of the data matrix and their length are stored in variables, and also other needed matrices are initialized with zeros. Hamming distances from all descriptors to all initial modes are computed to assign a cluster for each descriptor. After all data vectors have been assigned, the modes of the clusters are recomputed. This is done by keeping the number of ones and zeros in two matrices (Nones and Nzeros) for each attribute/feature of all descriptors of a cluster. In the two matrices, each row is a vector that keeps the number of ones or zeros for a cluster and each column represents an attribute/feature of the descriptors. Then, it is compared each value of one matrix and the other to know if there are more ones or zeros and define in this way the mode of every cluster.

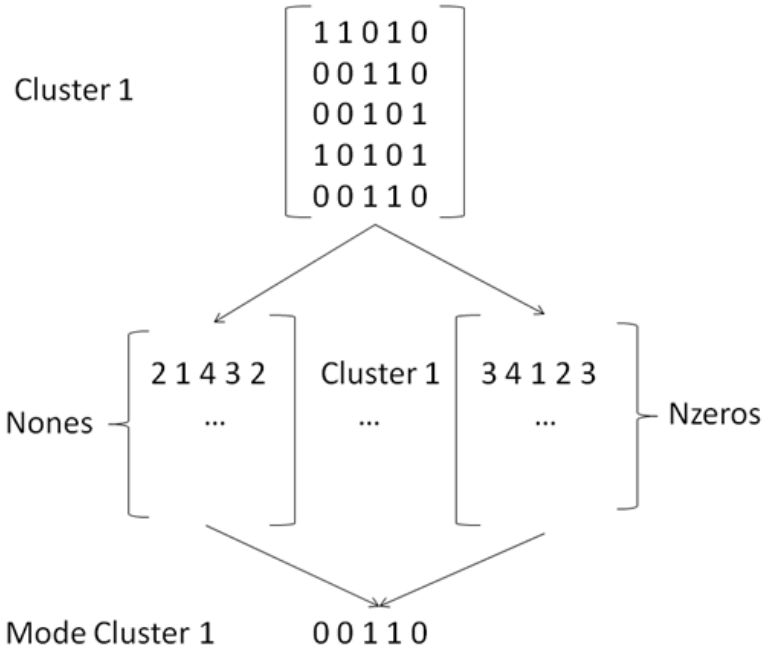

Fig. 9.6. Methodology for computing the mode of a cluster.

Once all data is assigned and all the modes are computed, an infinite while loop is entered to retest the nearest mode of all descriptors by using another time, the hamming distance (the minimum distance is kept to compute the cost afterwards). It only breaks the loop when there is not a change of class of a descriptor after all reassignments.

If one descriptor changes its class, the Nones and Nzeros matrices of previous assigned cluster and current cluster are updated and finally compared to update their modes.

To conclude the function, all distances from data to their cluster mode are added to compute the cost.

9.2.1.4. Evaluation and cost

The defined cost for kmodes algorithm is the sum of all distances from each descriptor to its mode class, so it is the sum of all the hamming distances. This cost is known as the intra-class variance and should be minimized, which is the optimization problem of the kmodes algorithm: for partitioning a set of n objects described by m categorical attributes into k clusters S_1, S_2, …, S_k:

$$\text{Minimize} \sum_{i=1}^{k} \sum_{X \in S_i} d(X, Q_i) \qquad \text{(Eq. 9.4)}$$

Where Q_i is the mode of cluster S_i.

Another metrics to evaluate the clustering result are the distance between modes, the interclass variance (should be maximized), minimum distance between descriptors (data points) of different classes and a lot of other index like Davies-Bouldin index or Dunn index.

9.2.1.5. Optimal k – Elbow method

The Elbow method is a methodology used to decide the optimal number of clusters k to use in a kmeans algorithm for clustering data. It is the more used method for deciding the k in kmeans algorithm. For kmodes algorithm, there are not officials and specifics methods for deciding the optimal number of cluster k, so it is done an analogy of the Elbow method for kmodes algorithm.

The elbow method plots the value of the cost function produced by different values of k. As it has been described in 9.2.1.4, the cost function of kmodes algorithm is the sum of the distances (hamming distances) from each data point to its mode. In the case of kmeans, the cost used is the SSE (Sum of Squared Errors).

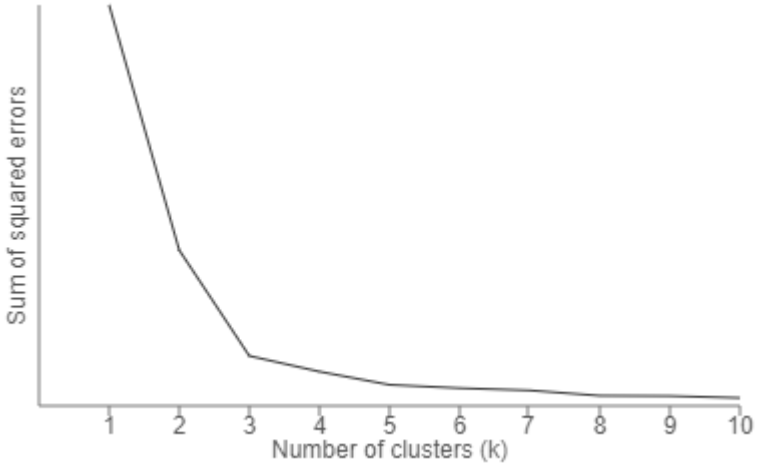

Fig. 9.7. Example of Elbow method: cost according to number of clusters

Figure 9.7 is an example of a chart of the SSE according to the number of clusters k. If the line chart looks like an arm, then the "elbow" on the arm is the value of optimal k. The idea is that we want a small cost but a not too big number of clusters because it decrease the computational speed and interclass variance, and increase computational cost. The cost tends to decrease toward 0 as it is increased k (the cost becomes 0 when k is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the center of its cluster). So the goal is to choose a small value of k that still has a low cost, and the elbow usually represents where the decrease starts to have diminishing returns by increasing k, that means the cost starts decreasing linearly.

Note that in Figure 9.7 is represented the elbow chart showing the SSE after running k-means clustering for k going from 1 to 10. It can be seen a pretty clear elbow at k = 3, indicating that 3 is the optimal number of clusters.

9.2.2. Wiener Transform

In this method [4] it is used the Wiener transform as a preprocessing step to convert binary vectors into real numbers vectors and then, apply conventional clustering algorithm as kmeans.

The Wiener transformation is a linear statistical transformation that considers the neighbors of each vector element. The Wiener filter is proposed by Norbert Wiener and its syntax is Y=WT(W,pq,noise) for two dimensional vectors (usually images). The input X is a two dimensional matrix and the output matrix Y is of the same size.

WT uses an element-wise adaptive wiener method which is based on the local neighborhood of each element. Adaptive wiener method means that it considers the sliding window concept, where a fixed size ones matrix, which is used to find the sum between nearby elements. The center element of sliding window is kept on the element to be considered on the vector. Then, for each element of the input matrix the sliding window elements are multiplied by input matrix elements and the result is added. Wiener estimates the local mean ω in equation () and variance σ^2 in equation () for each element using the following equations.

$$\omega = \frac{1}{pq} \sum_{n_1,n_2 \in \eta} X(n_1, n_2)$$

(Eq. 9.5)

$$\sigma^2 = \frac{1}{pq} \sum_{n_1,n_2 \in \eta} X^2(n_1, n_2) - \omega^2$$

(Eq. 9.6)

Where η is the p-by-q local neighborhood for each element in the input matrix X and pq is the size of the sliding window. WT then creates a element-wise Wiener filter Y using these estimates and it is given in equation (3.7).

$$Y(n_1, n_2) = \omega + \frac{\sigma^2 - \rho^2}{\sigma^2}(X(n_1, n_2) - \omega)$$

(Eq. 9.7)

In the case that the difference between the vector element's variance and p2 is negative means then its mean (ω) is directly taken as Y:

$$\text{if } (\sigma^2 - \rho^2) < 0$$

(Eq. 9.8)

$$Y = \omega$$

Wiener Transformation can be applied to all data types which are represented in numerical format. Here the binary data is pre-processed by transforming into real using the Wiener Transformation. The transformed data is clustered using the above mentioned K-means

algorithm.

This method is discarded because at a practical point of view, the binary vectors are categorical numbers with two categories 1 and 0, and the result of WT applied to binary vectors are categorical numbers with 4 categories: 0, 0.33, 0.66 and 1. This is the case of a row or column vector where the size of neighborhood and the element itself is 3. For that reason, is considered better to apply directly a categorical numbers clustering algorithm.

9.3. Experiment: recognition of kitchen objects

The real application is made with four kitchen objects: kh7, soap box, popcorn box and a tomato bottle. These objects are selected because they have some text, shapes and not uniform color that make them have singular points (features), they also have steady colors (do not change with illumination) and also because they are quite planar objects which is a good characteristic for describing keypoints. It is taken the tomato bottle, which is not a planar object with faces to check this kind of objects with BRIEF descriptor. All images are taken by hand and with the Pocophone F1 mobile phone.

As for the “Selection Methodology of the best Extension”, all the code is developed in Matlab and as it has been selected in chapter 8, it is used the YCbCr extension of the BRIEF descriptor that is the one that gives the best recognition rate but it is a little bit slower than the other ones. In all this section, this YCbCr extension is referred as BRIEF descriptor only.

The application is divided in three main Matlab codes, one for the Bag Of Features (BOF) creation, the second one for the training and validation of a Support Vector Machine (SVM) and finally several test codes.

For testing the algorithm, the following flow chart has to be carried out. The inputs and outputs of the codes are also represented:

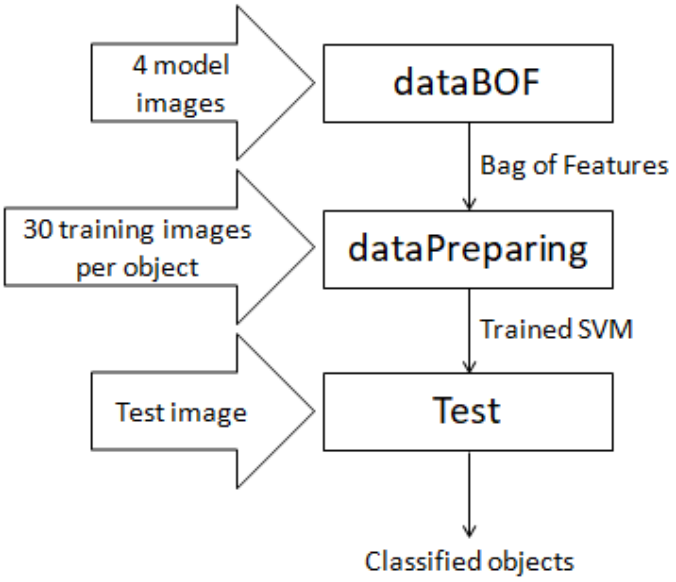

Fig. 9.8. Flow chart for testing the BOF model for object recognition.

The database of the four objects is composed of 30 images per object taken inside the kitchen and with artificial light. For the Bag Of Features it is used one model image per object, for the training the 30 images of each object, and finally, new test images are taken to evaluate the classifier.

For the BOF, training and some of test images, it is also taken a photo of the background that will be used to compute a difference of images between the image with the object and the background. This is done to be able to isolate the object from the background.

9.3.1. Bag Of Features design (BOF)

The Bag Of Features or Bag Of Visual Words is developed with one image per object. This model image is one of the 30 of the database, in particular, it is the first image obtained, where the object is visualized in frontal position and in a distance of 25 cm:

Fig. 9.9. Model object images for the BOF creation.

Fig. 9.10. Model objects images for the BOF creation.

With this configuration, the object has to be in frontal view in the test images to be able to recognize it. The classifier will support variations (object rotated some degrees for instance) depending on the training images.

First of all, the difference of the background and each object image is done to analyze the part of the image where object appears and to avoid obtaining features from the background.

Once it is obtained the image difference, it is binarized by a threshold value: if one of the channels of the RGB difference image has an intensity value over a given threshold, the pixel of the binary image is set to 255, which is the one value of the binarized image. It is not needed to put a condition to set the zero value to a pixel because the binary matrix is initialized with all zeros.

Fig. 9.11. Left image difference and right binary image.

As it can be seen in Figure 9.11, there are some noisy pixels that do not belong to the object, and other object pixels that are not set to 255. For this reason, the following morphological operations (used as Matlab functions) are applied in this order: bwareaopen(), imdilate(), imfill(),bwareaopen() and imerode():

- bwareaopen(): sets a 0 in areas smaller than a given threshold introduced as an input to the function.
- imdilate(): used to make the “ones” areas bigger by using a morphological structuring element.
- imfill(): fills image regions and holes.
- imerode(): used to make the “ones” areas smaller by using a morphological structuring element.

The last erode operation is used to avoid consider features that are in the borders of the object by later deleting the features that are not in the ones area (eroded object area). These features are not desired because they can vary their description depending on the background pixels.

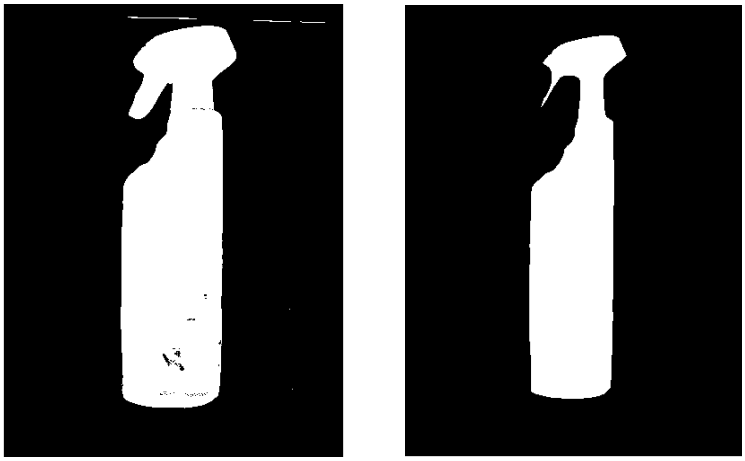

Fig. 9.12. Kh7 left binary image and right eroded binary image.

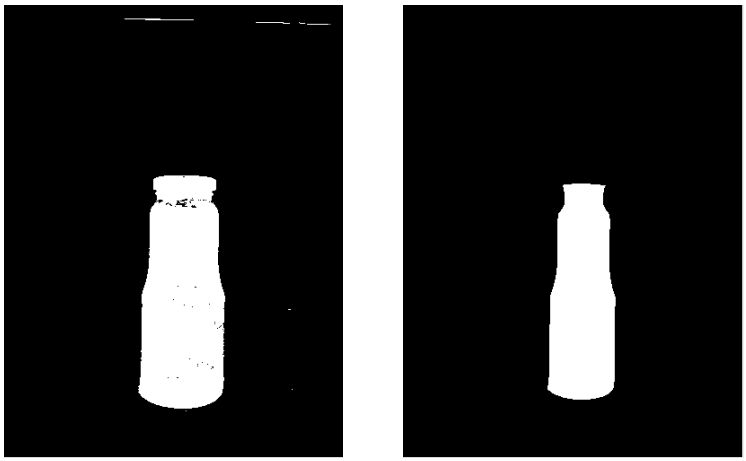

Fig. 9.13. Tomato bottle left binary image and right eroded binary image.

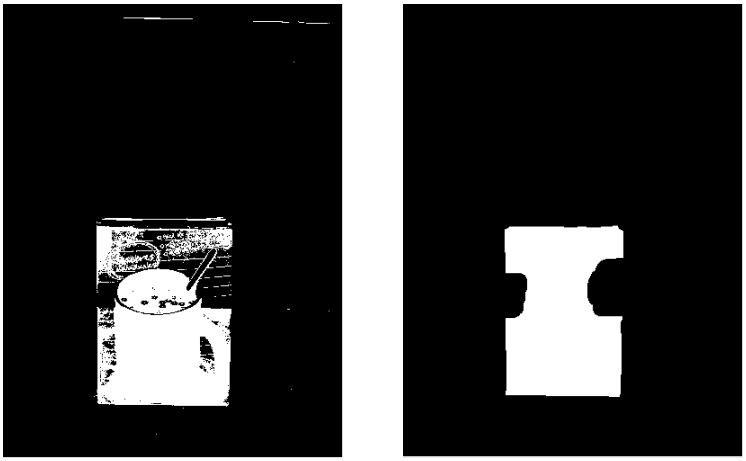

Fig. 9.14. Soap box left binary image and right eroded binary image.

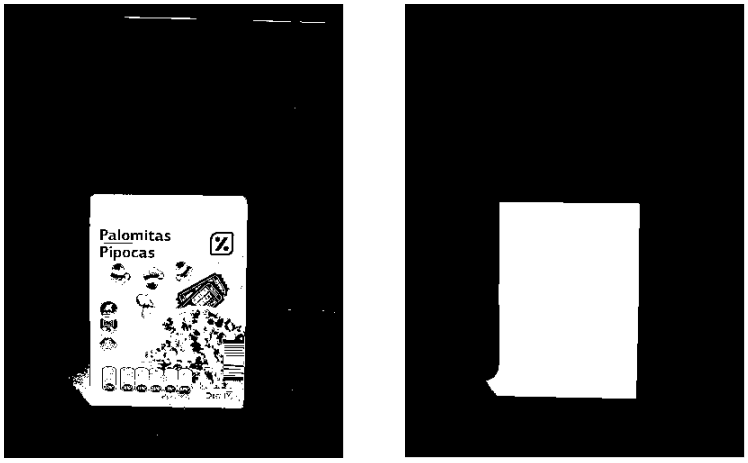

Fig. 9.15. Popcorn box left binary image and right eroded binary image.

After the pre-processing part, the detection and description of features is done. First of all, some variables and parameters of the BRIEF descriptor are initialized: two matrices to store the YCbCr and grayscale images, a vector with all zeros called descriptors which is the one used to concatenate later all the BRIEF descriptors according to each feature point of each object image (for using it to make the clustering, it is made in this way because the number of keypoints per object is different and unknown), and all the BRIEF parameters explained in section 4, that takes the optimal values given in the original paper of the descriptor. The random pattern generation for distances and channel selection of the binary tests is fixed to be the same for all the algorithm.

Then, it is defined a “for loop” to make the computations four times, one per object. First, the YCbCr and grayscale images are obtained. Then, the detectFASTFeatures Matlab function is used to detect the Fast keypoints for each object. As the output of this function gives the

coordinates as x and y (column and row), it is necessary to exchange the two columns to have y and x which is row and column. Now, for every point is tested if it belongs to the ones area of the eroded object binary image and the FAST keypoints that are out of this area are deleted. It is shown the grayscale image with the final keypoints detected:

Fig. 9.16. Kh7 and tomato bottle with FAST keypoints.

Fig. 9.17. Soap and popcorn boxes with FAST keypoints.

Finally, as it is needed a matrix with all the descriptors of all objects to later make the clustering, it is computed the new_descriptor matrix of BRIEF vectors with the function of YCbCr BRIEF descriptor. This new matrix of BRIEF descriptors has all descriptors of the feature points of one object and it is concatenated with the previous matrix of objects descriptors called descriptors, which at first is a vector of zeros as it has been commented, but later it accumulates all the descriptors of the objects. This way of grouping all the descriptors is not fast because the matrix of all the descriptors can not be initialized with a matrix of zeros to preallocate the memory needed, due to the already mentioned fact that the number of descriptors per object is different and unknown. It is also necessary to delete the first row of zeros to conclude the creation of the database of descriptors for the clustering to create the BOF.

With all the descriptors in the same matrix, now the clustering is performed with the kmodes algorithm. It is used a k (number of modes) of 60, which is selected based in the Elbow method:

Number of modes k	Cost
10	68646
20	58818
30	53315
40	50518
50	48725
60	47132
70	45802
80	44472
90	43758
100	42799

Table. 9.1. Cost values according to the number of modes.

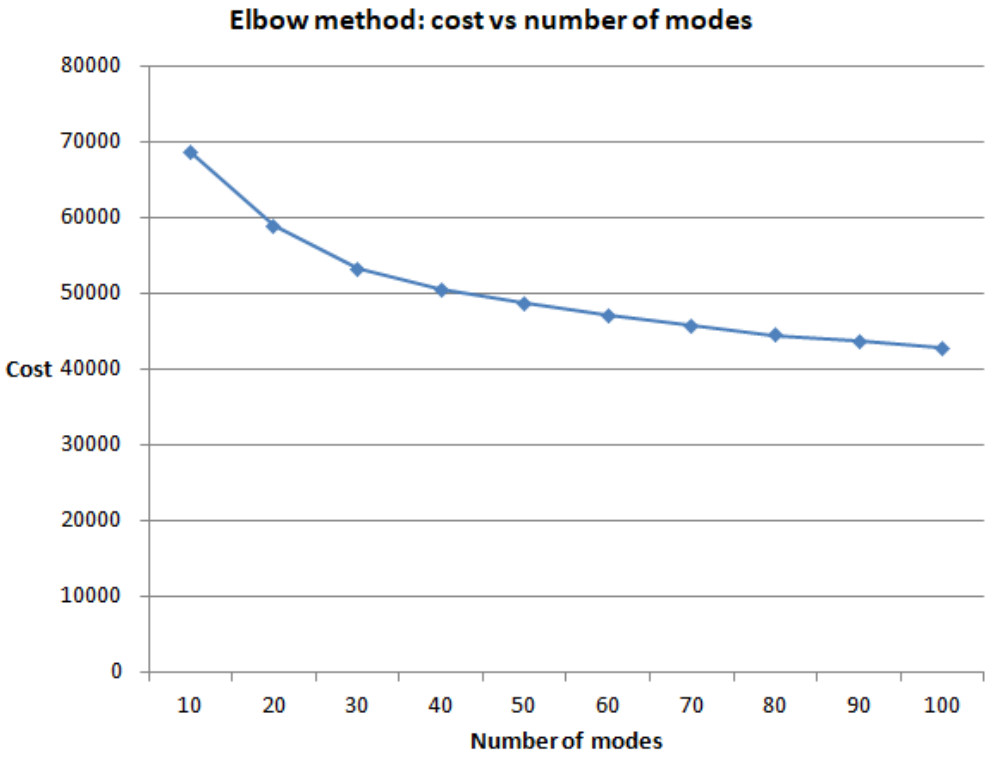

Fig. 9.18. Cost according to the number of modes chart.

As it has been explained in chapter 9.2.1.5., the elbow method compares the cost (intraclass cost obtained as an output of the manual implementation of kmodes) of different values of number of modes. In this case, it has been compared ten modes from 10 to 100 in steps of ten. As the cost is smaller, the recognition rate of the final classifier will be bigger but the speed decreases with the increase of the number of modes.

In Figure 9.18, the evolution of the cost according to the number of modes is represented. The cost decreases when the number of modes increases which is the true behavior of clustering methods. In this case, the elbow point is not too clear, can be clear that it is between 40 and 70 where the decrease step starts to saturate. It can be seen that in 40 clusters, the cost function starts to decrease linearly, but using this k in tests of chapter 9.3.3., the results are not good at all. Observing the graph, from ten to twenty, there is a big decrease of 9828 but then, as the number of modes is increased, the decrease step of the cost is smaller. This decrease step is the slope of the function between two values of number of modes and it is the derivative of the cost vs number of modes function. As the elbow point is not so clear, the slopes are computed and the derivative of the cost according to the number of modes is represented:

Number of modes range	Cost derivative
10-20	-9828
20-30	-5503
30-40	-2797
40-50	-1793
50-60	-1593
60-70	-1330
70-80	-1330
80-90	-714
90-100	-959

Table. 9.2. Cost derivative values according to the number of modes.

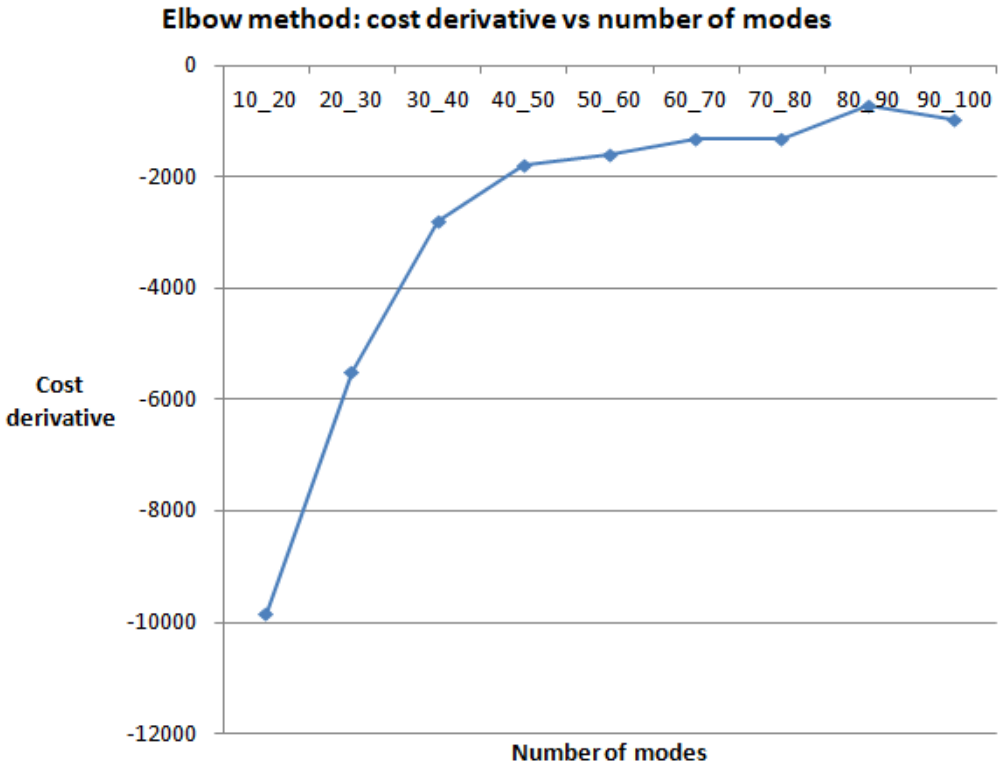

Fig. 9.19. Cost derivative according to the number of modes chart.

In Figure 9.19, the negative slopes of the decreasing steps can be seen very clear. At first, the slope is very negative but in the range 20_30 modes, the negative slope decreases a lot. Then, linearly it starts decreasing the negative slope until the range 60_70 modes. From 60_70 to 70_80 modes, the negative slope is maintained, and from 70_80 to 90_100 it starts fluctuating by increasing and decreasing the negative slope. Now, it is clear that the value where the cost starts to saturate is for 60 modes. So it is used 60 modes for the Bag of Features model.

The kmodes algorithm gives the cost, modes and index (class of each feature point) for the clustering algorithm. In the experiment case, the modes output is a matrix of mode vectors of 60 x 128, that are 60 modes and 128 is the mode length (same as the BRIEF descriptors).

9.3.2. Classifier: training and validation of a SVM

Once the Bag of Features (or Bag of Visual Words) is built, it can be used to obtain a BOF histogram as a feature vector per object to train a classifier.

30 images per object have been used as training images and two images of background. These two background images are necessary because a new background has to be obtained when the camera is moved. In this particular case, every time new training images are obtained with the camera in different position, the background without the object is captured. For the training of the 4 objects selected, one background is taken for kh7 and popcorn box and another for soap box and tomato bottle, as the camera was not in exactly the same position.

The training images are taken with the object in front view and some rotation in the vertical axis. It is also varied the distance from the camera to the object, which is in a range of 24-28 cm.

Fig. 9.20. Popcorn box training images.

The code that performs the preprocessing, training and validation of the classifier is explained bellow:

First of all, the workspace is cleared except the modes matrix, obtained from the kmodes function. This is done to free memory and make the code more efficient. After that, the difference of images is computed between each training image and its background. This is done since it is desired to train features that belongs to the object and are not disturbed by the background.

The same morphological operations than the ones applied on the Bag Of Features preprocessing part are applied on all the training images to isolate the objects and to later discard the features that do not belong to the object, are in its borders or near (as they are affected by the background).

Now, a “for loop” goes through all training images and for each one the following operations are done:

- The FAST keypoints are detected and filtered by belonging to the eroded object or not as in BOF code.
- The BRIEF descriptors of one object features are computed and stored in a matrix called descriptor.
- The descriptor matrix of one object is converted to a histogram of the Bag of Features. This is done by looking for the nearest neighbour of each BRIEF descriptor against each BOF mode, using the hamming distance. Each descriptor increases by one the BOF mode that belongs to. In this way, it is created a histogram that keeps the number of descriptors that belongs to each mode.

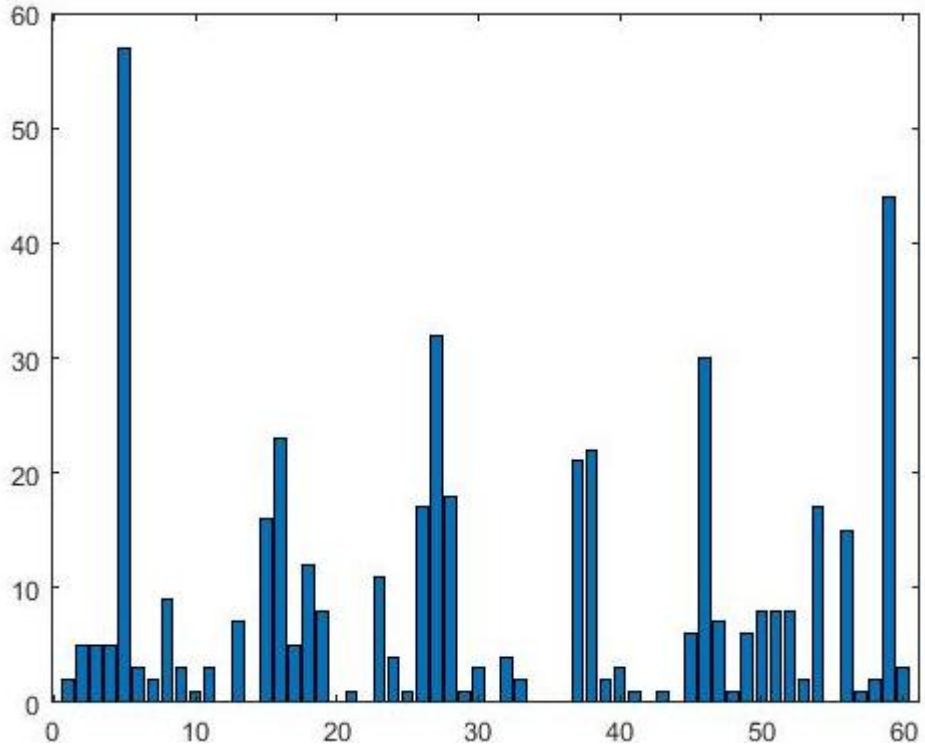

Fig. 9.21. Soap box BOF histogram.

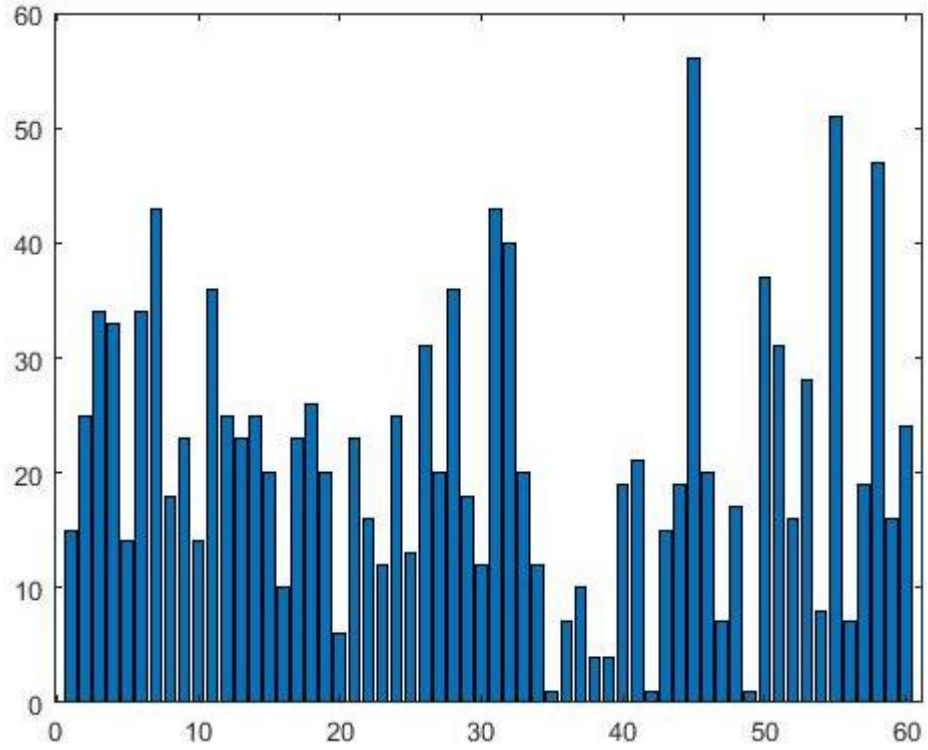

Fig. 9.22. Popcorn box BOF histogram.

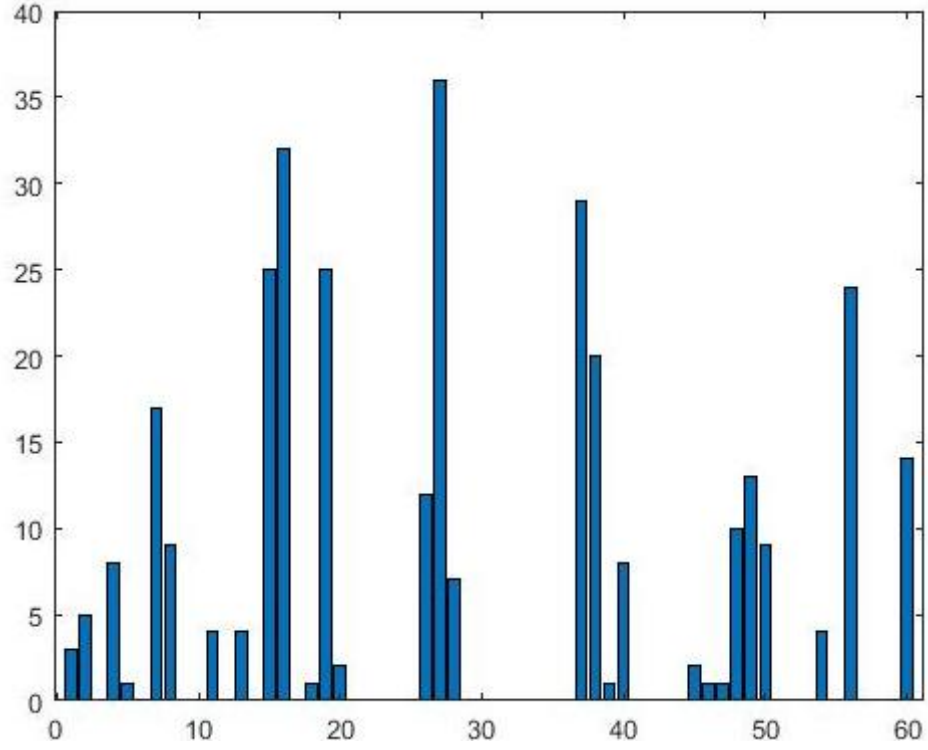

Fig. 9.23. Kh7 BOF histogram.

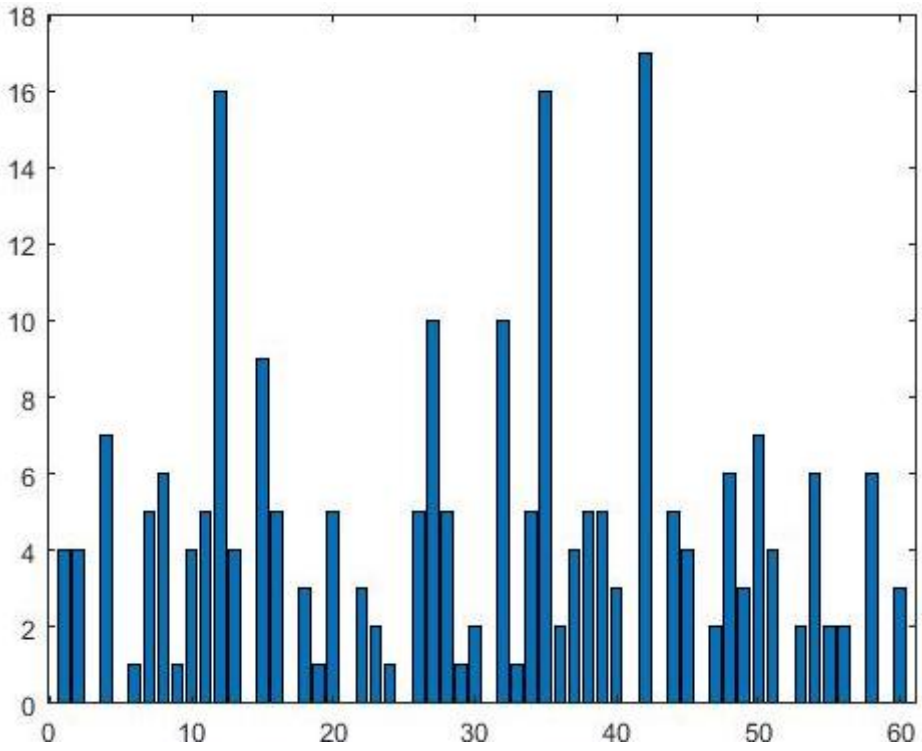

Fig. 9.24. Tomato bottle BOF histogram.

Finally, it is obtained a matrix of 120 x 60 called histograms, where the 120 BOF histograms of the training images are stored and have a length of 60 bins which are the modes.

After that, the labels (1 kh7, 2 popcorn, 3 soap and 4 tomato) of all training images are defined in a column vector that is concatenated to the last column of the histograms matrix, and the histograms and labels are stored in a new matrix called data. To conclude, it is used the Classification Learner Matlab application to obtain a classifier:

- **Open the application from Matlab.**

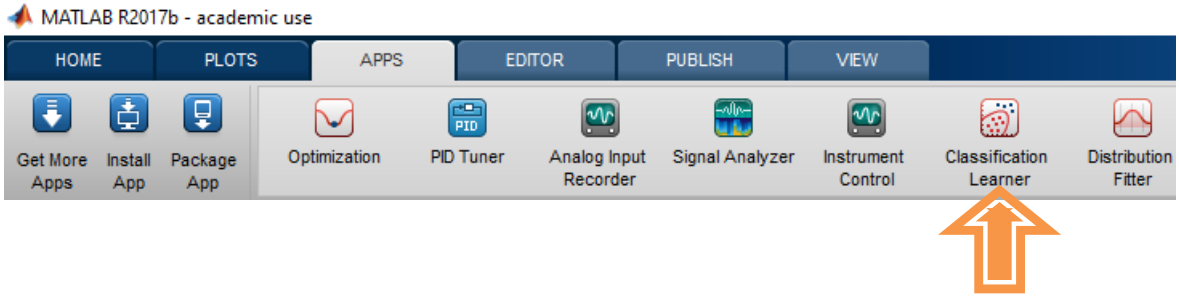

Fig. 9.25. Classification Learner app opening.

- **Upload data from workspace**: select New Session From Workspace.

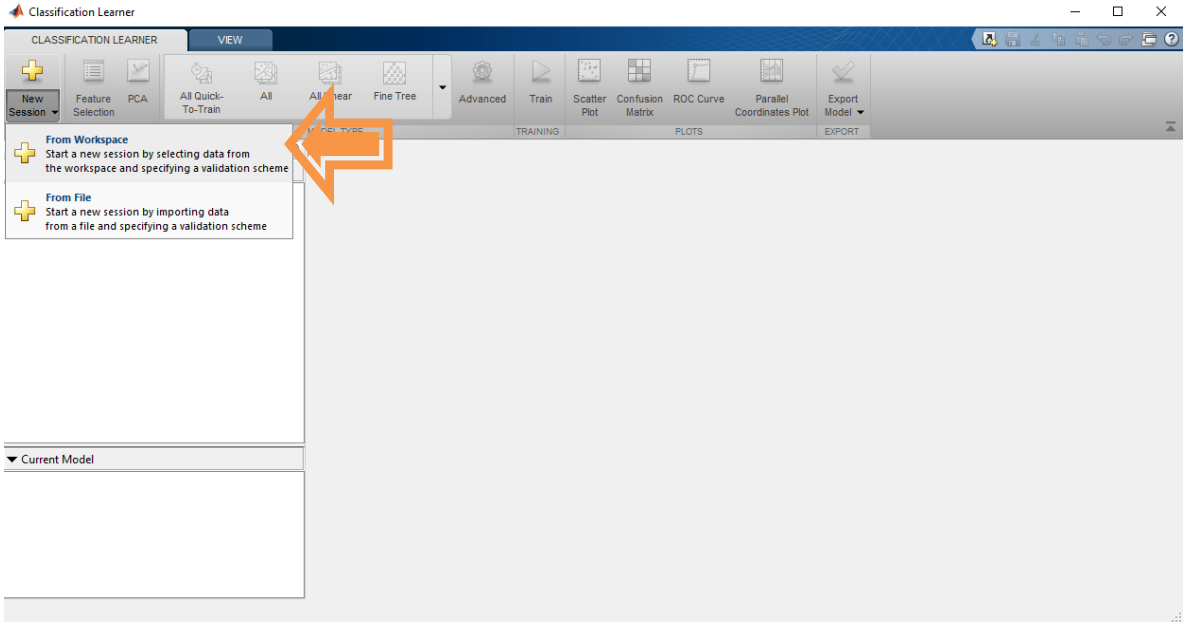

Fig. 9.26. Starting a new session.

In Select Input list, select the data matrix.

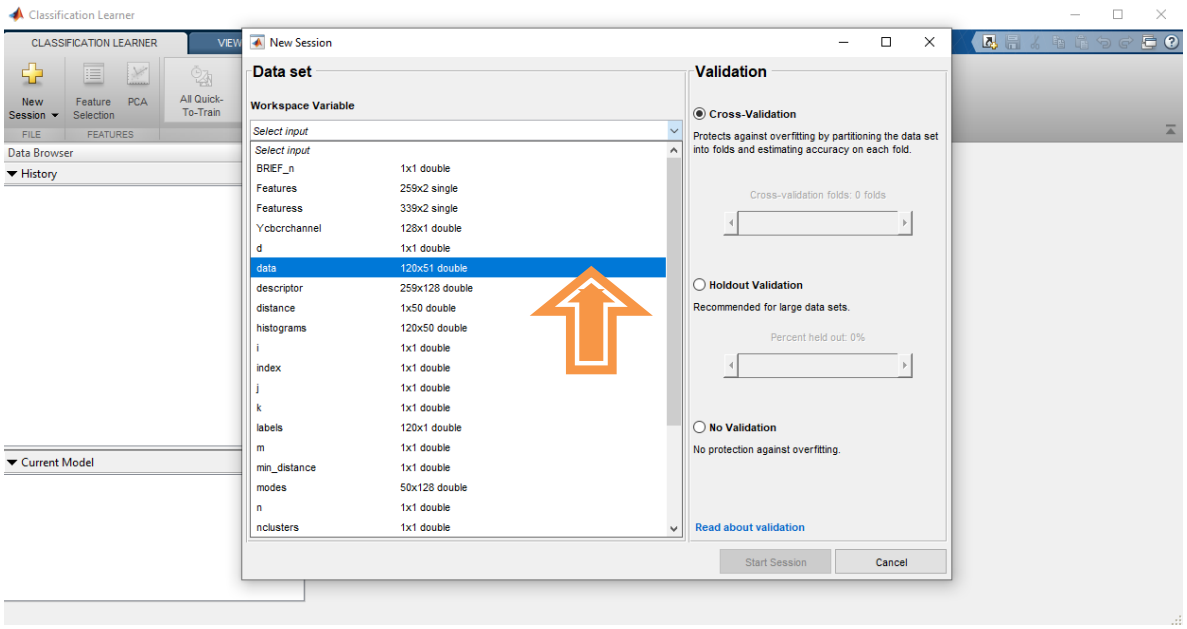

Fig. 9.27. Selecting the training data.

It is used the Cross-validation method to check the accuracy of the classifiers in the validation stage. As the application says, it protects against overfitting by partitioning the data set into folds and estimating accuracy on each fold. Select Start Session:

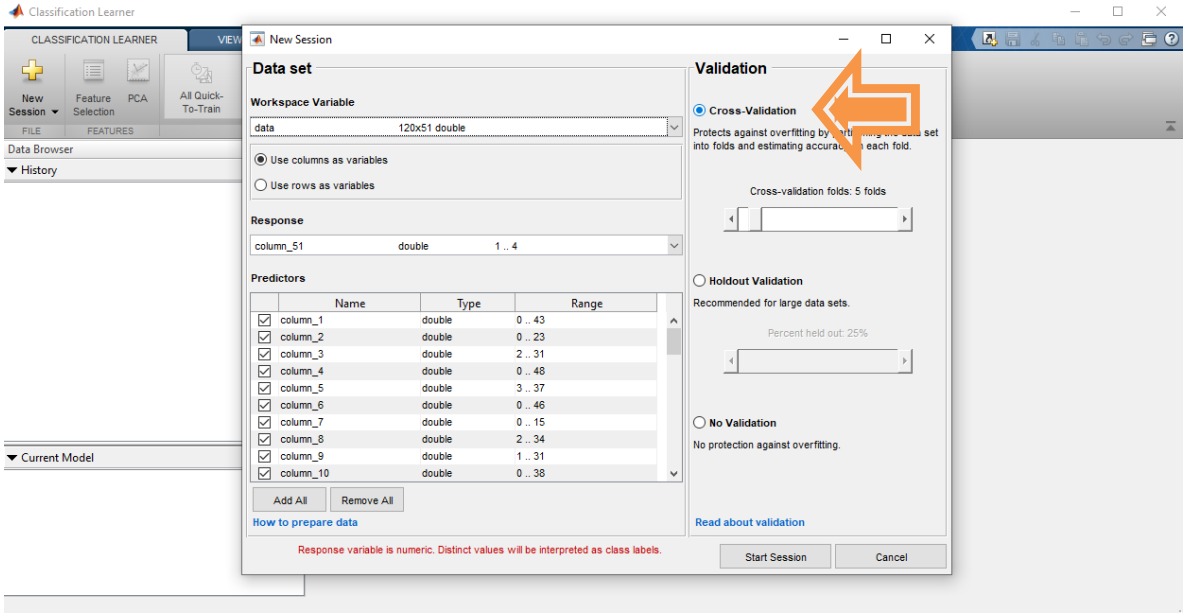

Fig. 9.28. Selecting the cross-validation option for validation stage.

- **Train all possible classifiers**: in Model Type select All and click in Train.

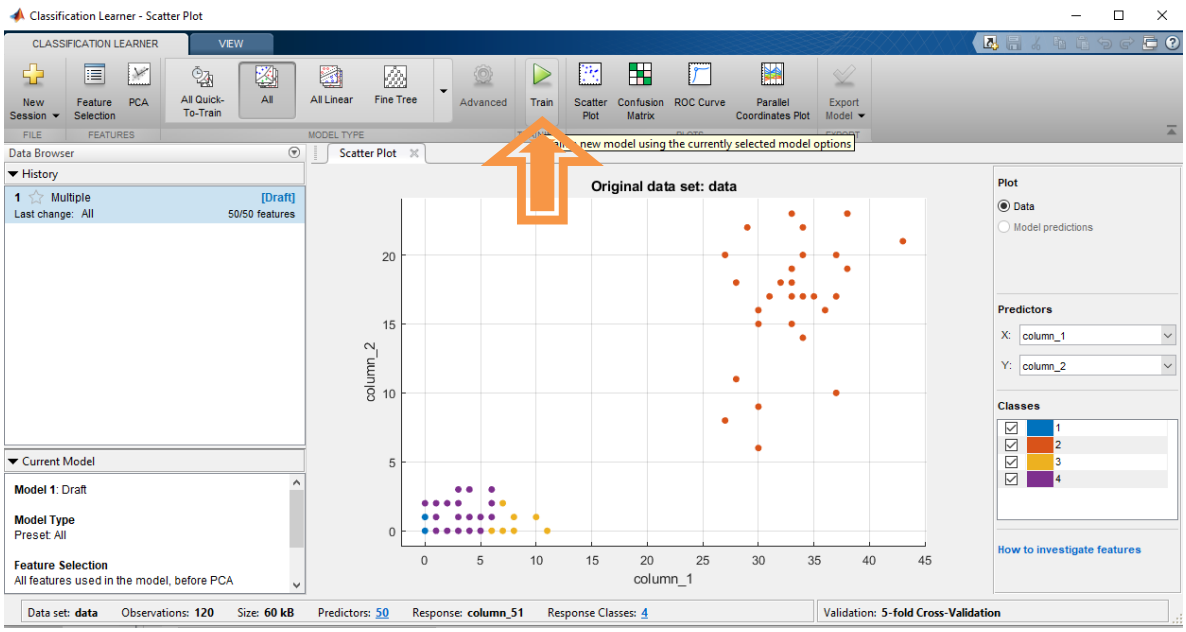

Fig. 9.29. Starting the training of all types of classifiers.

- **Select the best one and generate code**: the accuracy results are shown. Select the best classifier and export code by clicking on Export Model and Generate Code:

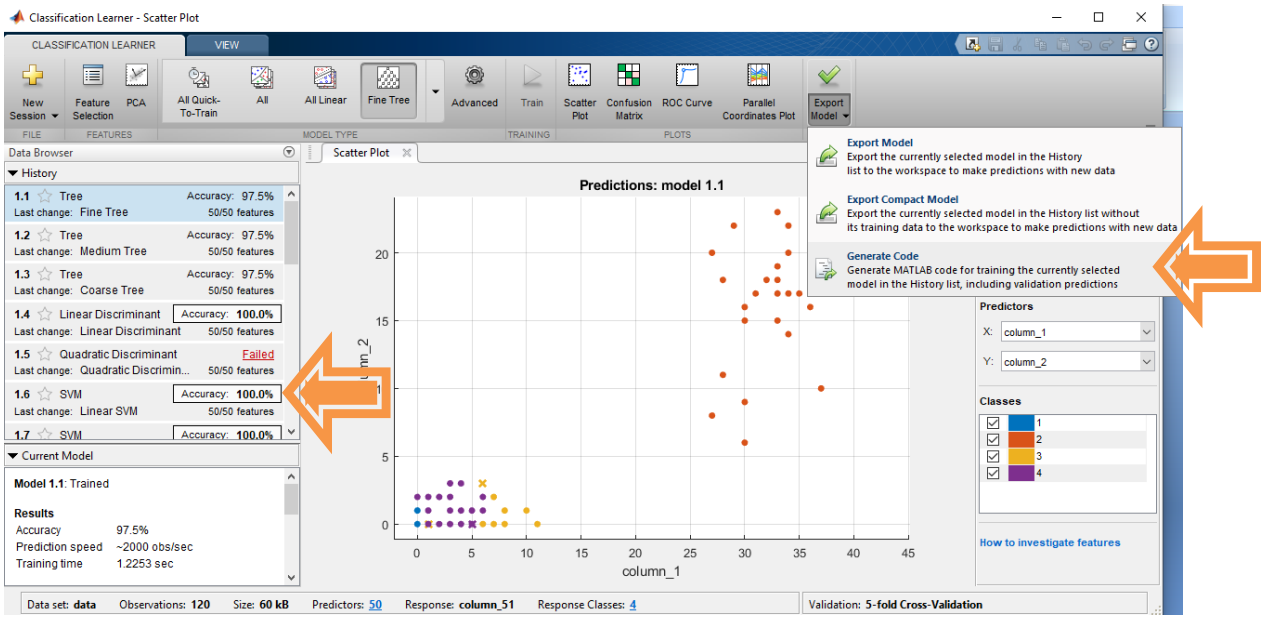

Fig. 9.30. Generating the classifier code.

So it is generated a code that implements the training and validation of a linear Support Vector Machine, that it has an accuracy of 100 %. It has been tested with different number of modes for the BOF histograms (range 40-60 for chapter 9.3.1. conclusions of the Elbow method) and the one that gives better results in the tests and it is also quite fast is with 60 modes.

9.3.3. Object recognition tests

Some different tests have been carried on to test the recognition rate and speed of the developed classifier in different conditions. At first, a basic test is made to check basic conditions. After that initial test, rotation and scale variance are tested to determine the ranges where the classifier is able to recognize the objects. Then, changes in illumination are checked to test if it is robust to light variations. As final tests, it has been tested the efficiency of the classifier when some parts of the objects are occluded.

9.3.3.1. Basic test

For the first test, it is used an ideal environment, with the same background and illumination as the training images that were taken in a kitchen room and with artificial light. All objects

appear in the same image and the distance from camera to objects is 29 cm, which is a distance a little bit bigger than in training images. The objects are not resized, so in this manner, the robustness to scale variation is also tested.

Fig. 9.31. Basic test image.

To analyze each object separately two approaches can be used:

- **Difference of images**: as it has been made in BOF and training codes, it consists in making the difference of the background and test image. The main drawback is that the camera has to be in exactly the same position and orientation in the two photos. Another disadvantage is that usually the objects shadows appears when making the difference and can make the objects split difficult in the binarized image. The good think is that the background is not considered and the only obtained features are in the object.

- **Sliding window**: this technique is based in the application of the recognition process in a portion of the image by usually using a square window, and then moving the window throw the entire image for finding the object. A big drawback is that the object is not separated from the background, so noisy feature points in the background can appear and affect the recognition accuracy. In this case, it is very important to have a quite homogeneous background. The localization of the object is a segmentation problem that has not been faced in the current project as the part of interest is to check if the classifier is able to recognize the object and not to localize it. For that reason it is used a “simulated sliding window”, which is simply to define a window in the object area for applying the classifier directly in the area of interest where the classifier has to recognize the object.

Difference of images:

In this test code, it has been used the difference of images technique, so it is obtained an image of the background. As in training images, the difference of images is performed and the result is binarized.

Fig. 9.32. Binarized difference of images of Basic test image.

As it can be seen in image 9.31, the shadow of objects appears and they attach the objects between them. Furthermore, it is also shown that for soap box the difference of images is not good at all because the color of the box and the wall of the kitchen are quite similar. After the binarizing the difference of images, the same morphological operations as in training images are used but with different values to be able to separate the objects and then, their bounding boxes are stored.

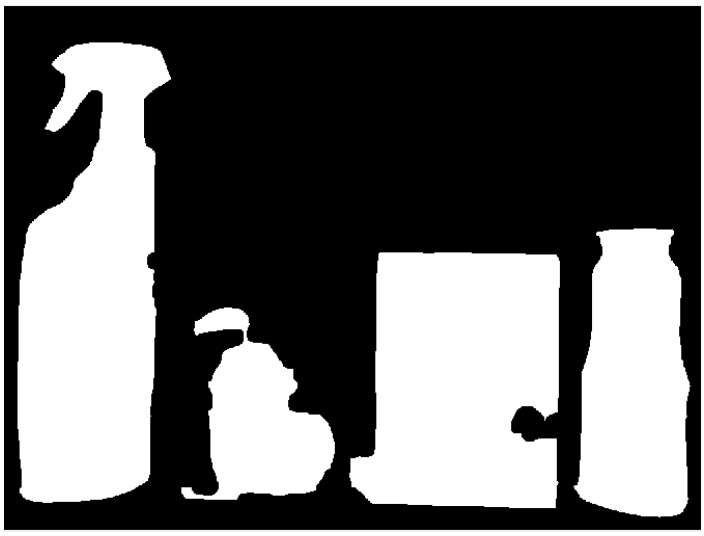

Fig. 9.33. Eroded Binarized difference of images of Basic test

In figure 9.32, appears the objects separated but it remains a little bit of shadow. The objects have been also eroded to ensure that points in borders or near are not considered. For soap box object, the binary object image has eroded a big part of the object due to the similarity with the background and the morphological operations. This is another big disadvantage; so the background has to be different than the objects.

Now, the same steps as for training images are developed to obtain classified objects: fast feature points detection, deletion of points in borders or near (the ones that are out of the binary object image), BRIEF descriptor of remaining points, transformation to BOF histogram and prediction of class with the linear SVM classifier.

The results are very good, the classifier is able to recognize the four objects, so the recognition rate is 100%. As they are in same conditions than in training images, the histograms are very similar to the training images and different from each other:

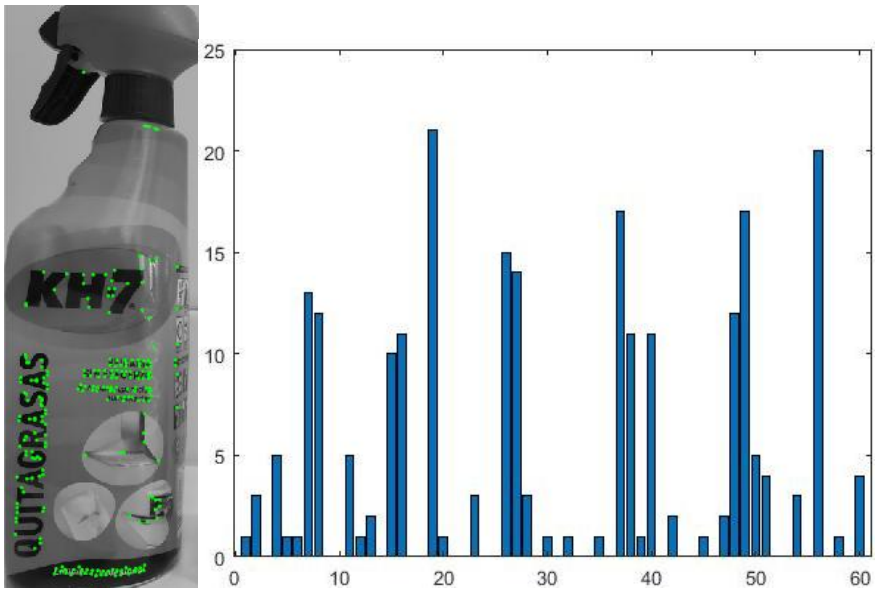

Fig. 9.34. Kh7 with FAST points and its BOF histogram.

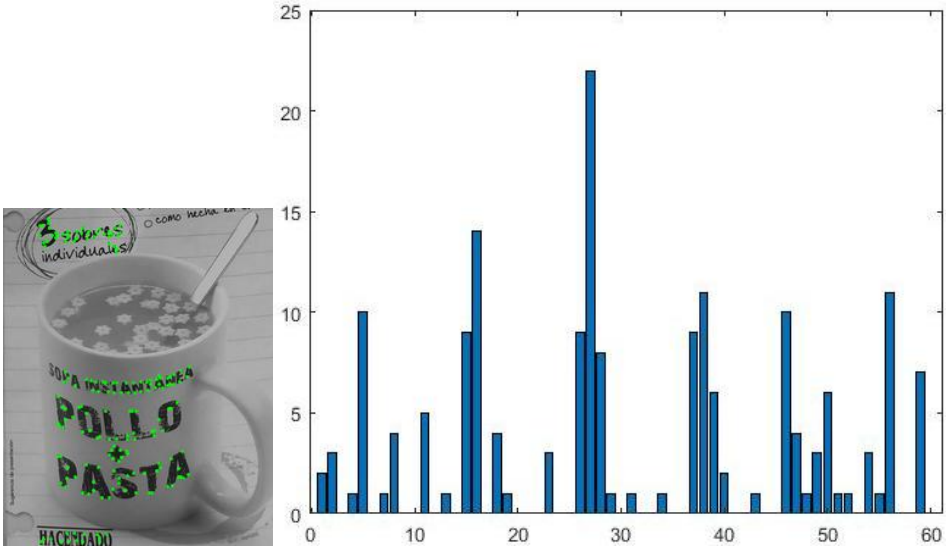

Fig. 9.35. Soap box with FAST points and its BOF histogram.

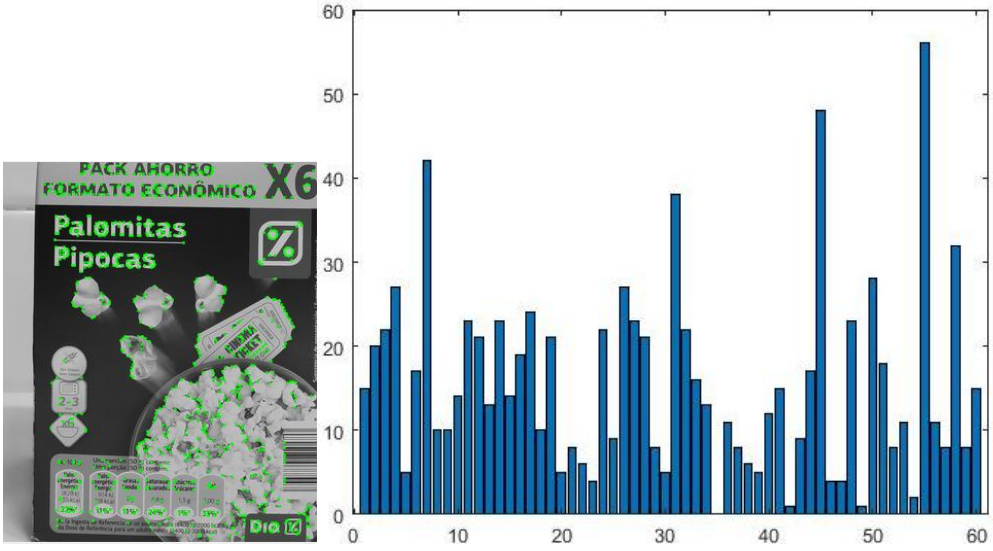

Fig. 9.36. Popcorn box with FAST points and its BOF histogram.

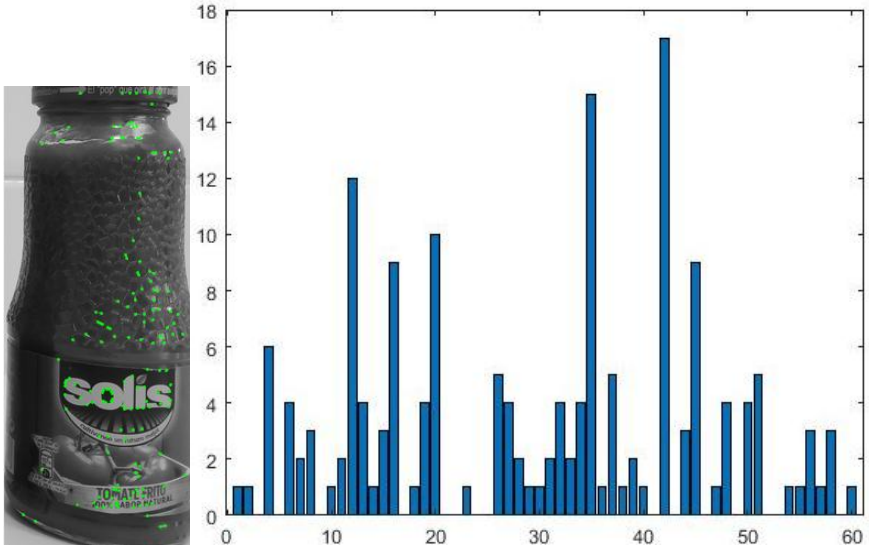

Fig. 9.37. Tomato bottle with FAST points and its BOF histogram.

The results of FAST keypoints detected appear in the grayscale object images because this detector works with the images in grayscale.

Simulated Sliding window:

Now, it has been implemented a combination of both methods, simulated sliding window and difference of images. In this test is used binarization of difference of images to take features inside the objects and a simulated sliding window to avoid eroding too much the objects to separate them.

In this case, the same values than in training images are used for morphological operations. The result is the objects eroded but joined by their shadow:

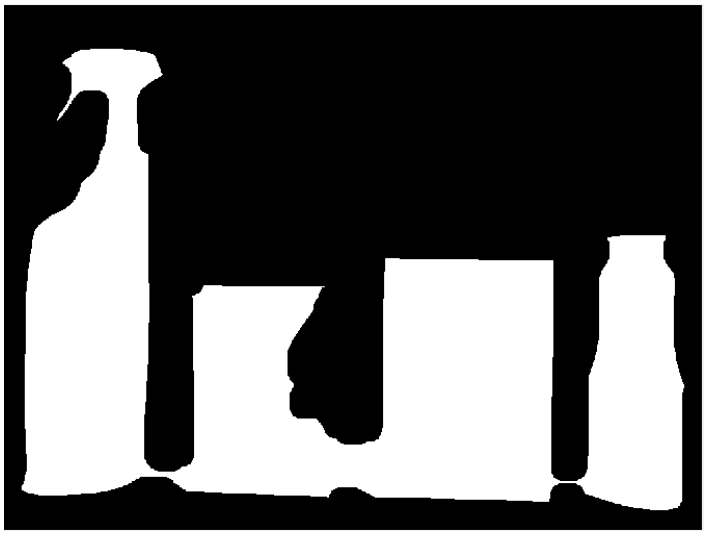

Fig. 9.38. Eroded Binarized difference of images of Basic test without splitting the objects.

As it can not be used the bounding box, a sliding window is simulated by defining the zones of the objects. This is done because the application of sliding windows techniques is not the goal of this project. In a real sliding window technique, a sliding window would be defined and it would be moved through all image, or different sliding windows would be defined with different sizes, one per object or one per soap box, popcorn box and tomato and another one for kh7 which is the different one.

The rest of the code is the same than for the difference of images code. The results are also very good since the classifier is able to recognize all the objects.

Fig. 9.39. Window for soap box with FAST points.

As it is shown in figure 9.38, the sliding window is defined incorporating some background on the object but it does not affect the recognition process because it is a quite uniform background and it does not detect FAST points in there.

As a conclusion, the basic test checks the recognition of the objects when they appear in the same test image. The same conditions than in training images make it ideal environment to recognize them, but the objects are a little bit farther than in training images.

9.3.3.2. Rotation change

The main check of this test is the rotation supported by the classifier, but furthermore, the kitchen light was changed to a LED fluorescent lighting which light up more. It is also added noisy pixels from background and borders of object because it is not used the difference of images technique.

The objects are at 30 cm distance and rotated some degrees in vertical axis:

Fig. 9.40. Rotation test.

The test code uses a sliding window technique which is simulated as in previous tests to analyze the object patch only. As in this case, there is not difference of images, there are also not preprocessing steps. The code starts reading the image and defining the BRIEF descriptors parameters: 128 for descriptor length, 11 x 11 the window size, the same fixed random numbers generated by a normal distribution with 0 mean for point coordinates and channel selections.

Now, the same steps as for training images are developed to obtain classified objects except the deletion of points out of the object or near/in the borders of the object: fast feature points

detection, BRIEF descriptor of FAST points, transformation to BOF histogram and prediction of class with the linear SVM classifier.

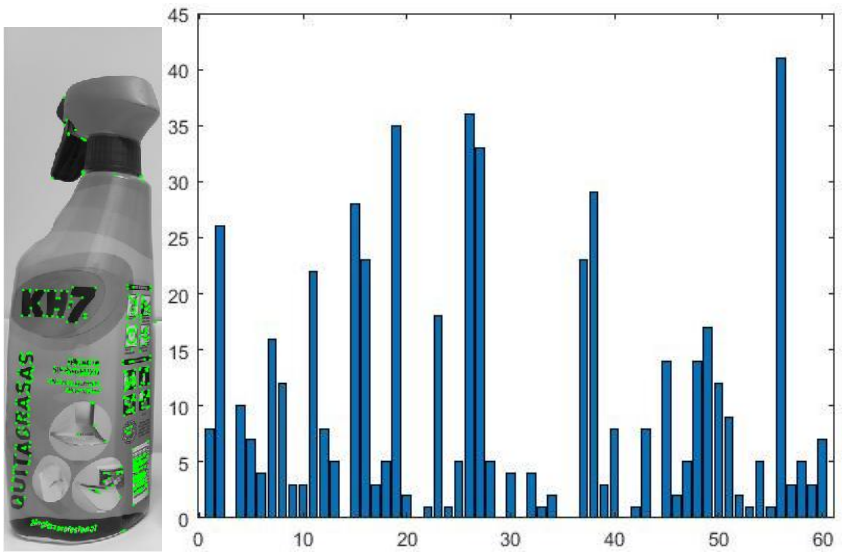

Fig. 9.41. Kh7 FAST points and BOF histogram.

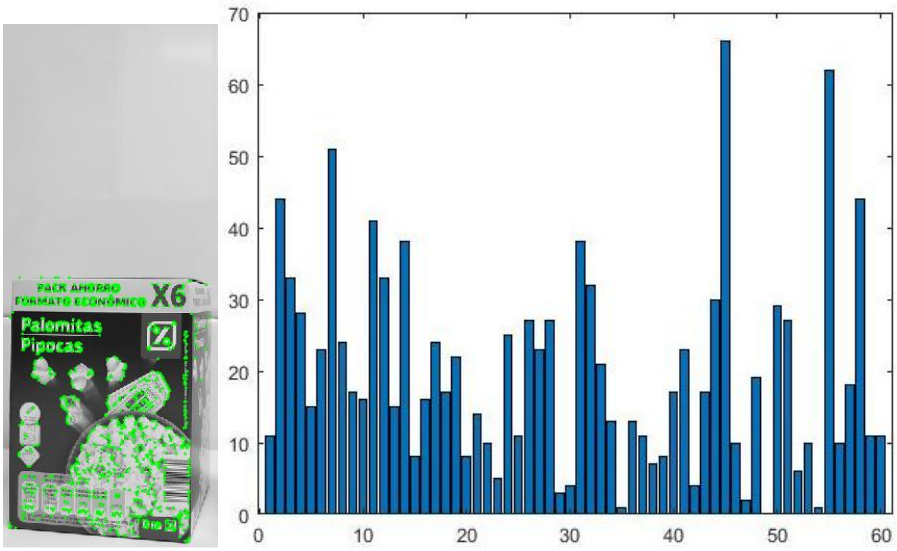

Fig. 9.42. Popcorn box FAST points and BOF histogram.

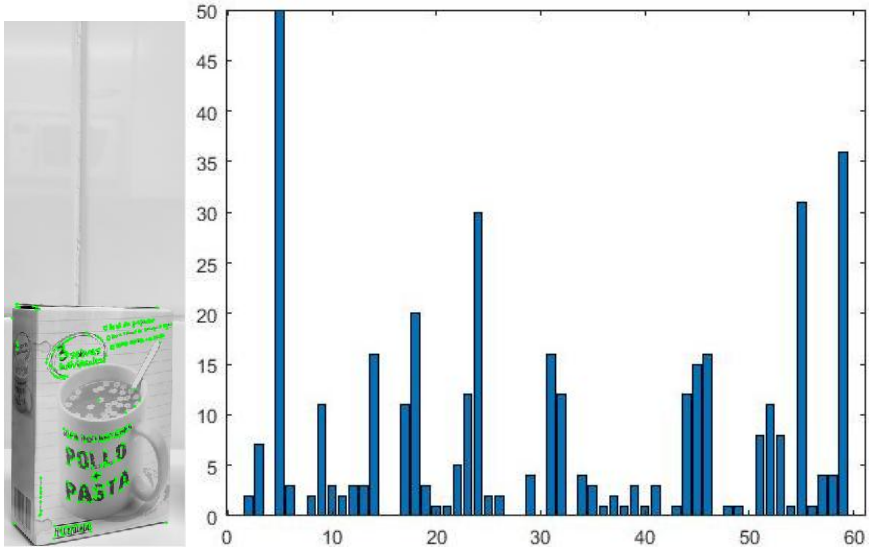

Fig. 9.43. Soap box FAST points and BOF histogram.

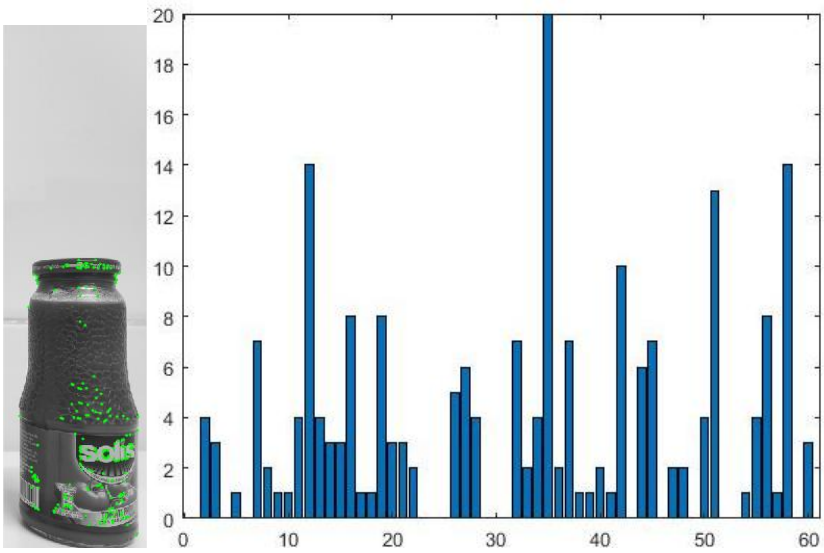

Fig. 9.44. Tomato bottle FAST points and BOF histogram.

The classifier predicts correctly the classes of popcorn box, soap box and tomato bottle. It fails with the kh7 by classifying it as soap box, because it is too much rotated, and it takes a lot of FAST points in the lateral view of the object as it can be seen in figure 9.40. In the other objects, the algorithm detects some points in lateral views but a quantity supported by the classifier, as it has been trained with the objects rotated a little. Objects images shows that the FAST detector does not detect FAST points in the background as it is quite uniform, but it detects some points in the borders of the object. These points are not desired since it considers the background when it describes the keypoint with the BRIEF descriptor, but the classifier supports these noisy points and they do not affect to its performance.

9.3.3.3. Scale change

In this subsection, the scale robustness is tested by locating kh7 and soap box at 23 cm, and tomato bottle at 32 cm (out of the range of the training images 24-28 cm). The object patches are analyzed without rescaling them, so in this manner, the scale range supported by the classifier is checked.

As in subsection 9.3.3.2, the illumination is stronger than in training images. Furthermore, noisy feature points can appear in background and near/in the borders of the objects because it is used the simulated sliding window technique instead of difference of images. In following images, it is shown the different illuminations for the scene:

Fig. 9.45. Scale test and comparative of the two types of illumination.

Then, in following images can be seen the FAST points detected in grayscale images, as it works in this color range:

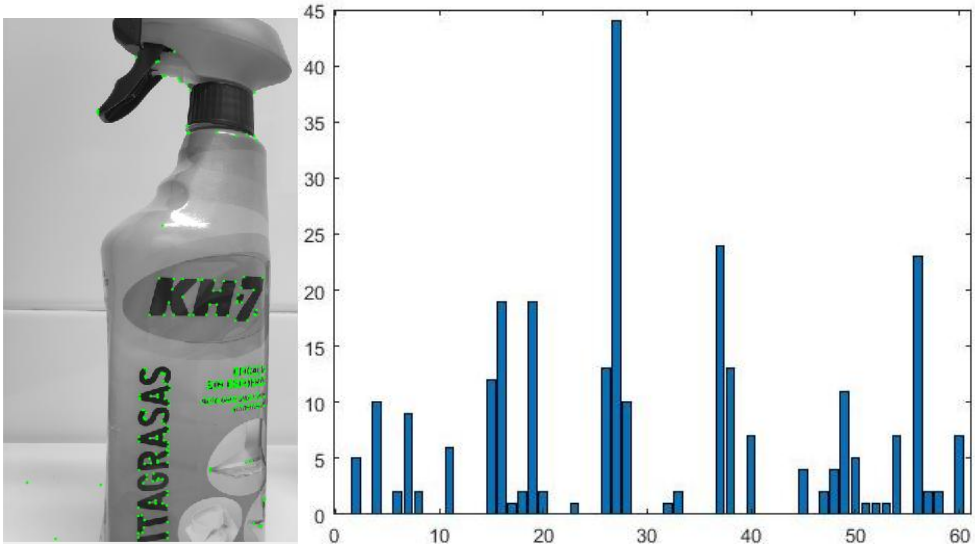

Fig. 9.46. Kh7 with FAST points and BOF histogram.

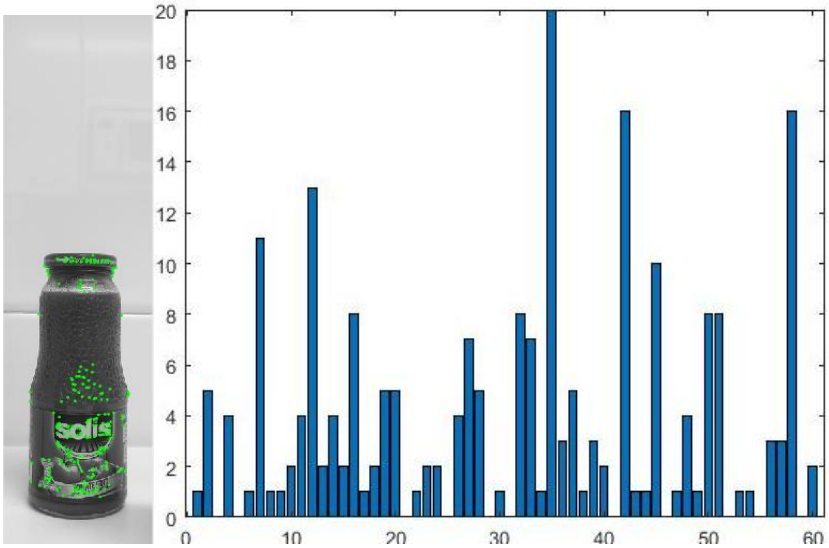

Fig. 9.47. Tomato bottle with FAST points and BOF histogram.

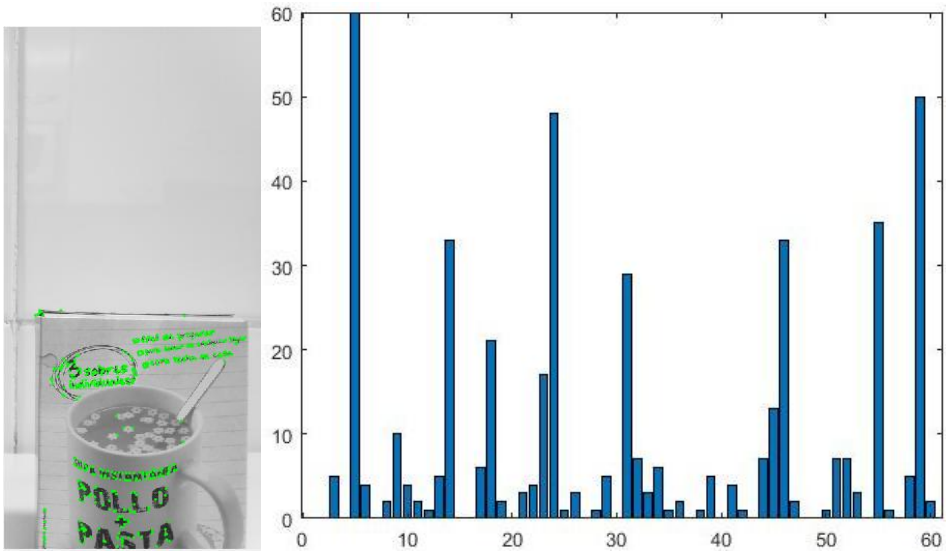

Fig. 9.48. Soap box with FAST points and BOF histogram.

The results of this scale test are very good, the classifier predicts correctly the class of all objects showing a good performance in the 23-32 cm scale range. It is also shown, the support of this change of illumination. BRIEF descriptor is not scale invariant itself as it computes differences of pixels given distances. In this case, the main thing that makes the classifier support some change of distance is that the objects are not very heterogeneous or color textured with little shapes. Another good thing done to support some variation of distance is that the classifier has been trained with some variation in distance, the objects were in a distance range of 24-28 cm.

9.3.3.4. Illumination change

Now, two changes on illumination are tested, one with poor natural light inside the kitchen room and another test with more natural light outside the kitchen. In this second case, the environment is completely different, with natural light and different background, and the objects are at a distance of 28 cm.

Fig. 9.49. Illumination test with objects outside kitchen with natural light.

The background is outdoor, not so uniform and could be also variable due to wind movement. As it can be little movements of the environment, camera or shadows; the difference of images is not a good technique because it can appear some borders on the moving parts.

Fig. 9.50. Binarization of image difference of outside illumination test.

As it can be seen in figure 9.50, the wind causes movement and the camera had also a minimum movement, this causes the appearance of borders with the objects.

It is decided to use the combination of difference of images and simulated sliding window. It is also simulated the sliding window as in basic test because the part of interest is the application of the classifier in the object area. After applying morphological operations the result is the following one:

Fig. 9.51. Binarization of image difference of outside illumination test.

In figure 9.50, remain shadows and some parts that were not deleted with morphological operations. The objects are also joined, so with difference of images, it is avoided to take FAST points in the borders or near the object, and now with simulated sliding window, it is possible to analyze the objects separately. If the sliding window had not been simulated, it would have been applied the classifier in the parts that are not deleted and do not belong to objects. In this case, the classifier would have been trained with a class of no one of the objects, composed by images of the background parts that could appear after morphological operations. The same has to be done in the case of the use of sliding window technique without difference of images, since the window goes through all parts of the scene and not in all appears the object; so it has to be a trained class of no object.

In following images, the FAST keypoints detected are plot. It can be seen that appear some noisy points in the background that would be avoided defining the simulated sliding window (object patch) closer to the object, but in this way, it is checked how much affects the noisy points.

Fig. 9.52. Tomato and kh7 with FAST keypoints detected.

Fig. 9.53. Kh7 and popcorn box with FAST keypoints detected.

Finally, the last stages are done as for the basic test: FAST points detection, BRIEF descriptor for each keypoint, conversion to BOF histogram and apply linear SVM classifier. The results for this test are quite good, the classifier is able to recognize the objects despite having different light and noisy points, but in tomato-kh7 image, the classifier fails with kh7 due to the poor quantity of FAST points in object.

Now, for the first case where the objects are inside the kitchen without artificial light, it is tested the developed classifier. In these conditions, it is checked 2 objects and 4 objects. The test with 2 objects has the objects at a distance of 26 cm:

Fig. 9.54. 2 objects illumination tests inside kitchen with natural light.

So it uses only difference of images and all the steps of the object recognition process. As in basic test with the difference of images technique, stronger values of morphological operations are used to separate objects, and the result is quite good.

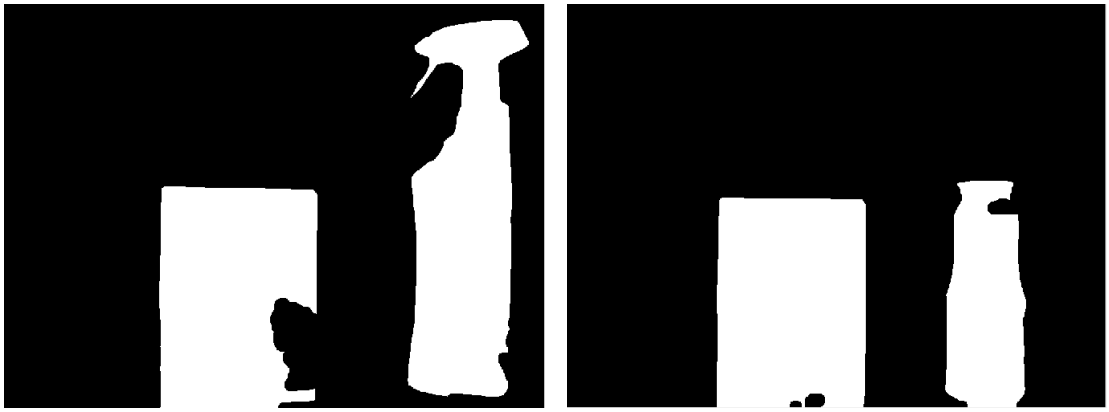

Fig. 9.55. Eroded binarized image difference of 2 objects illumination tests.

The final results are quite bad, and it only classifies well the kh7 by chance. The first problem is the detection of FAST points. For these conditions, the Matlab function detectFASTfeatures detects very few feature points in kh7 and tomato bottle objects. Apart from that, it behaves bad also with popcorn box despite detecting the usual number of keypoints.

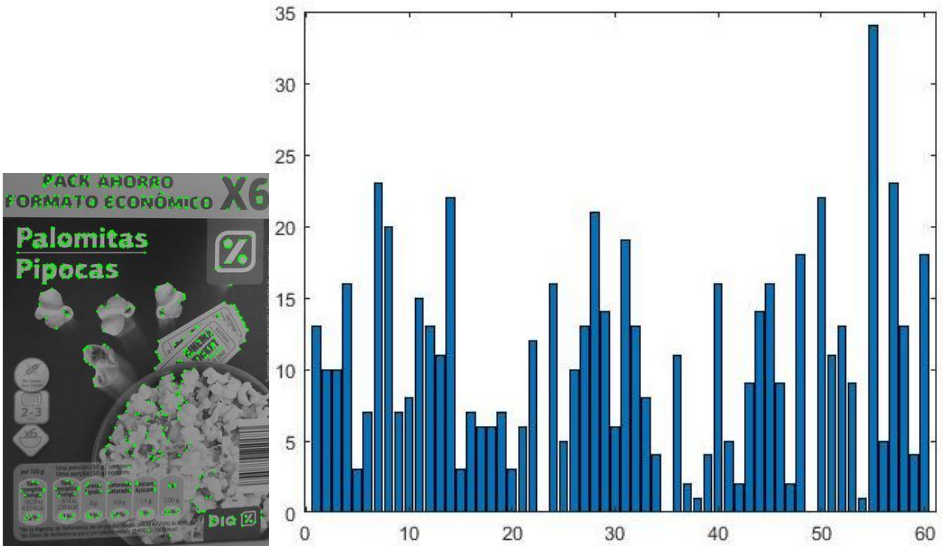

Fig. 9.56. Popcorn box with FAST points and BOF histogram.

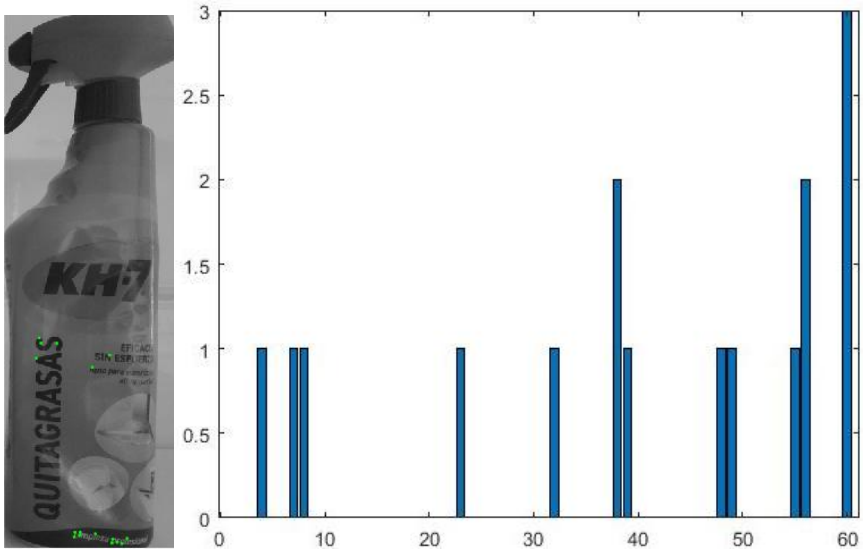

Fig. 9.57. Kh7 with FAST points and BOF histogram.

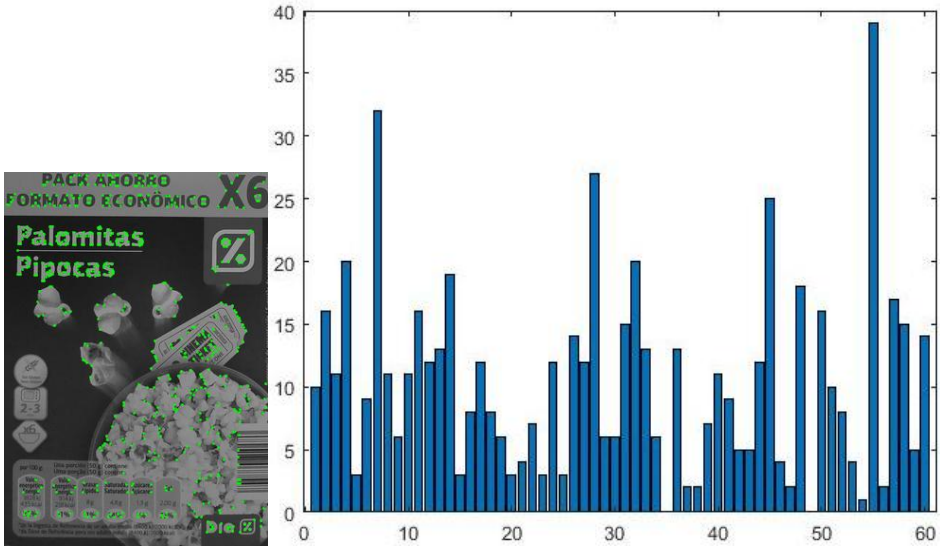

Fig. 9.58. Popcorn box with FAST points and BOF histogram.

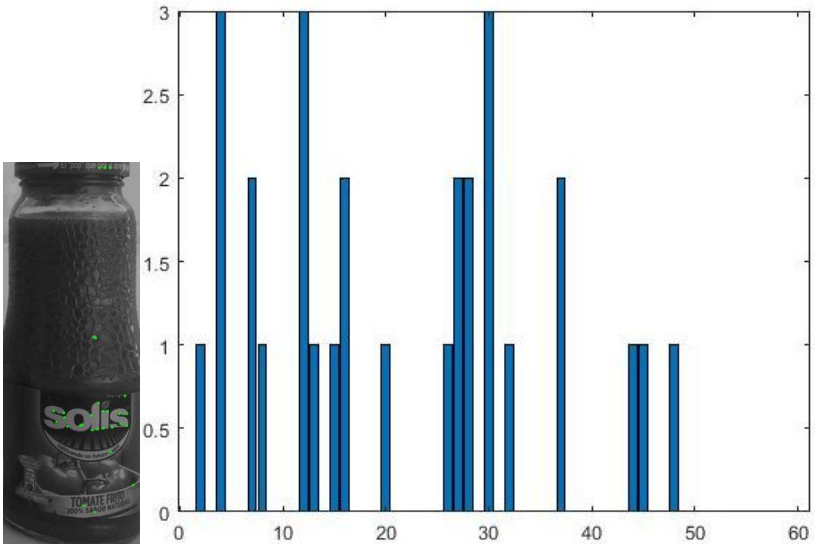

Fig. 9.59. Tomato bottle with FAST points and BOF histogram.

Same results are obtained when these conditions are tested with the 4 objects (at a distance of 29 cm) in the same scene. It is applied difference of images technique and morphological operations (with the same values than training part) and a simulated sliding window. The simulated sliding window defines the object areas and avoid eroding too much the objects for separating them.

Fig. 9.60. 4 objects illumination test with artificial light.

In the case of the tests where the objects are inside the kitchen but without artificial light, the results are bad, the classifier does not work well and it can not recognize the objects. This is due to the big change in the illumination, the poor natural light is very bad to detect the FAST points and to describe them.

9.3.3.5. Objects occluded

The final tests are designed to check the accuracy of the classifier when the objects have some part occluded. It is tested occlusion with a part out of the scene, occlusion by an object which is not from the database and occlusion by objects of the database.

The easiest test is the one that the objects have some part out of the scene. In this case it is used difference of images technique and follows the same steps explained in chapter 9.3.3.1. Kh7 and popcorn box are located at a distance of 26 cm:

Fig. 9.61. Occlusion test with some part of object out of the scene.

The results of this first occlusion test are quite good, the classifier recognizes both of the objects for left image and kh7 object in right image.

Now, it is used a pan to occlude some part of kh7 and popcorn objects. When an object occludes another, the one that is occluding is in front of the occluded one, so the objects of the database are also at a distance of 26 cm.

Fig. 9.62. Occlusion test with a pan as occluding object.

This test is more difficult because the algorithm gets keypoints of the pan object that can disturb the recognition process, and act as noisy features.

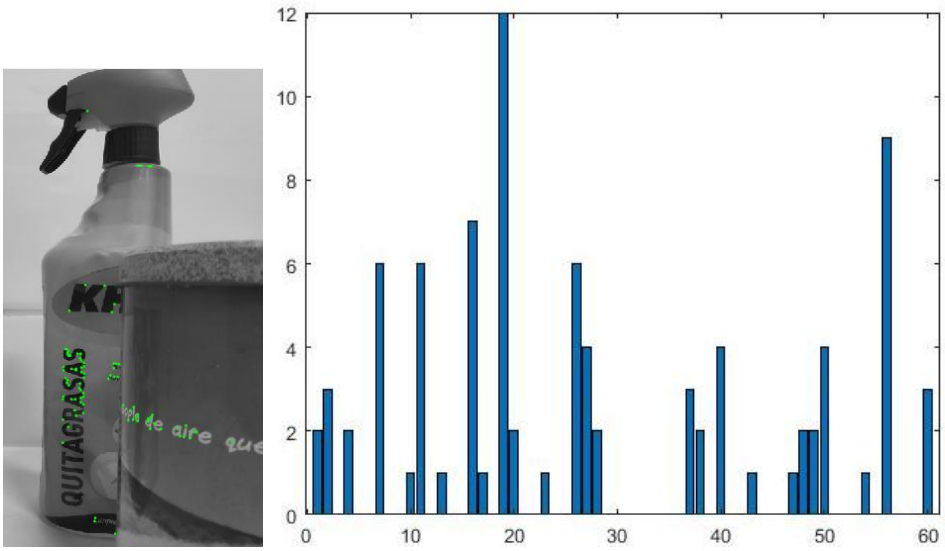

Fig. 9.63. Occluded kh7 with FAST points and BOF histogram.

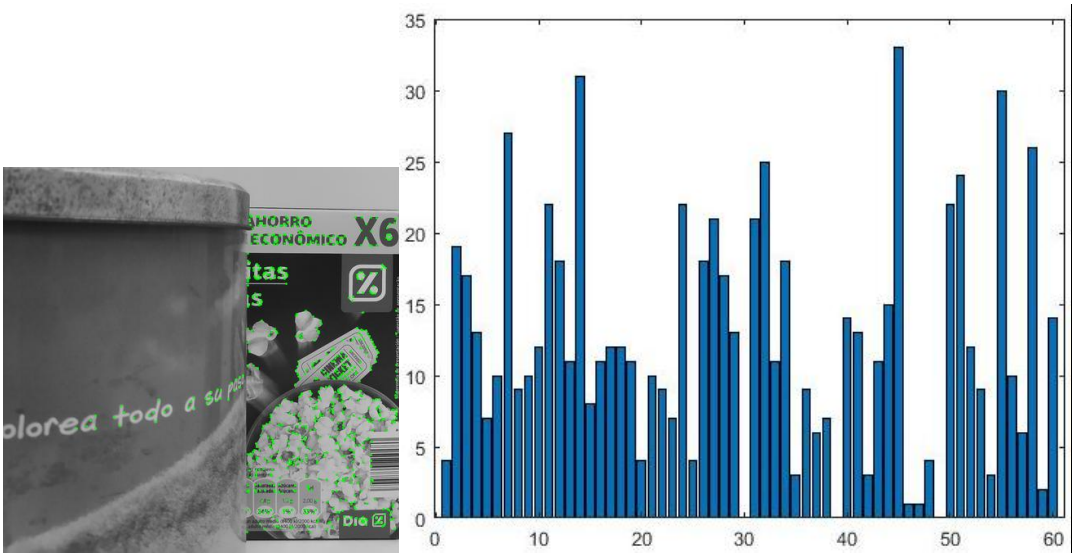

Fig. 9.64. Occluded popcorn box with FAST points and BOF histogram.

The results for this second test are also very good, it recognizes both objects although having an object with features in the front that can decrease the number of features detected in the object of interest (case of the kh7 object).

Finally, it is tested the particular case where the same objects of the database are being occluded between them. For this test, it is used tomato bottle, soap box and two different kh7. As they are in contact, it is not possible to separate them with difference of images and morphological operations, so it is also used simulated sliding window. The front objects are 25 cm away and the occluded ones at a distance of 30 cm.

Fig. 9.65. Occlusion test with database objects as occluding object.

The classifier only fails in soap box maybe because it is too much eroded with the morphological operations due to its similarity of its colors with the background. The different heads of the kh7 do not affect to the recognition process because is a zone where there are not a lot of keypoints:

Fig. 9.66. Kh7 objects with FAST points.

In general, for occlusion tests, the results are quite good. It can be confirmed that with local features techniques, the classifiers supports recognizing objects with occluded parts. On the other hand, the recognition rate depends on the parts that are occluded, if important keypoints are occluded, the classifier may not recognize the object but if the occluded part is uniform and has few or not strong enough keypoints, the classifier works well.

10. Budget

The current project is a research project so it only includes the software tool and the hours dedicated to develop the project. As the master final thesis has 12 credits, the total dedicated hours by the student are 360 hours. For the hours dedicated, it is considered the UPC specification for final thesis: 1 credit is equivalent to 30 hours.

To estimate the budget, two kinds of personnel services are considered: thinking engineer and developer engineer. The first one includes all the thinking and designing tasks of the project as for example BRIEF descriptor variants design and experiments setup. For the second kind of personnel service, it includes all the execution of the project, the programming, tests or technical documentation. The thinking and design tasks are about the 20% of the hours dedicated and the execution tasks as developer engineer are about the 80%. It is also considered the dedication of the TFM director, which is considered to be the 10% of the total hours dedicated by the student.

For determining the personnel costs of an engineer with master's degree education level (developer engineer service), it is considered the last approved Spanish BOE (January 18th, 2017) that establishes a salary of 23618,28 annual gross euros. Considering 230 working days (considering 23 holidays), gives a salary of 12,836 €/h. This is the wage of the engineer, but for the personnel services it is considered the cost of an engineering consulting which is considered to be 3 times the wage of the engineer. For the cost of the thinking engineer service, it is considered to be the double of the developer service. Finally, the TFM director salary is established as seven times the developer engineer wage.

Developer engineer = 3 x 12,836 €/h = 38,508 €/h.

Thinking engineer = 2 x 38,508 €/h = 77,016 €/h.

TFM director = 7 x 12,836 €/h = 89,852 €/h.

Concept	€/hour	hours	TOTAL cost in €
Thinking engineer	77,016	72	5545,152
Developer engineer	38,508	288	11090,304
TFM director	89,852	36	3234,672
TOTAL			**19870,128 €**

Table. 10.1. Personnel cost.

The software tool is Matlab R2017b and the following toolbox are necessary: Image Processing Toolbox and Computer Vision Toolbox. The software licenses and costs are for a complete year, but the real cost is considered for 4 months, which is the project duration.

Concept	€/year	Months	TOTAL cost in €
Matlab R2017b	800	4	266,66
Image Processing Toolbox	400	4	133,33
Computer Vision Toolbox	500	4	166,66
TOTAL			**566,65 €**

Table. 10.2. Software cost.

To conclude, the final budget of the project is shown in the table below:

Concept	COST in €
Personnel costs	19870,128
Software costs	566,65
TOTAL	**20436,778 €**

Table. 10.3. Project budget.

Conclusions

It is clear that the BRIEF extensions improve the original BRIEF in terms of recognition rate. As you increase the number of channels of a color space, the color information is bigger and the recognition rate increases but at the same time, the speed decreases. A big discovery of this project is that color space YCbCr works better than RGB because it has a quite bigger recognition rate and the speed is similar. For instance, with YCbCr BRIEF descriptor the recognition rate double improves but the elapsed CPU double increases.

Moreover, it has been tested the efficiency of new YCbCr BRIEF descriptor in object recognition applications with different environment conditions (basic conditions, rotation, scale, illumination and occlusion) where the results of experimental tests are quite good. It has been developed a new object recognition algorithm with a BOF model based in FAST keypoints and YCbCr descriptor which is a structure never used before.

As improvements of the current project, rotation and scale invariance can be added to new YCbCr BRIEF descriptor. This will make the descriptor more robust and completely invariant to scale and rotation. Although the descriptor will be more reliable, the speed will decrease, so the application defines the necessities of the descriptor giving more importance to speed or invariant to rotation or scale. Another improvement of the object recognition algorithm is to add all the views of each object to be able to recognize the object in front view but also in lateral and back views.

Acknowledgements

I want to thank my TFM director for his help and good advices in the execution of the project, and also, for giving me the motivation to do this project by introducing me the BRIEF descriptor in a lesson of the subject “Advanced Topics in Computer Vision” and by giving me the opportunity to contribute in the automated kitchen project providing a new object recognition algorithm with a new descriptor of local features.

Bibliography

References

[1] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. *BRIEF: Binary Robust Independent Elementary Features.* CVLab, EPFL, Lausanne, Switzerland.

[2] Michal Kottman. *The Color-BRIEF Feature Descriptor.* Faculty of Informatics and Information Technologies, STU Bratislava.

[3] Mahmoud Zidane. *BRIEF descriptor using FAST detector.* MATLAB Central File Exchange. Retrieved January 5, 2020. [https://www.mathworks.com/matlabcentral/fileexchange/66256-brief-descriptor-using-fast-detector, September 6th, 2019].

[4] Loraine Charlet Annie M.C. *Frequent Itemset Mining Based on Binary Data Clustering. Chapter 3: Binary Data Clustering Based on Wiener Transform.* DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING MANONMANIAM SUNDARANAR UNIVERSITY. September 2012.

[5] Zhexue Huang. *Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical* Values. Data Mining and Knowledge Discovery 2, 283–304 (1998).

[6] *Affine Covariant Features.* Includes database of original BRIEF paper. [http://www.robots.ox.ac.uk/~vgg/research/affine/, September 15th, 2019].

Complementary bibliography

- Frank Karstens. What is the RGB color model? The Basler Support Team. [https://www.baslerweb.com/en/sales-support/knowledge-base/frequently-asked-questions/what-is-the-rgb-color-space/15179/, September 14th, 2019].
- From Wikipedia, the free encyclopedia. RGB color model. [https://en.wikipedia.org/wiki/RGB_color_model, September 14th, 2019].
- From Wikipedia, the free encyclopedia. Color spaces with RGB primaries.

[https://en.wikipedia.org/wiki/RGB_color_model, September 14th, 2019].

- Intel. Developer Zone. Color Models. [https://software.intel.com/en-us/ipp-dev-reference-color-models, September 20th, 2019]
- PC Magazine Encyclopedia. YCbCr definition. [https://www.pcmag.com/encyclopedia/term/55147/ycbcr, September 20th, 2019].
- From Wikipedia, the free encyclopedia. YCbCr. [https://en.wikipedia.org/wiki/YCbCr#YCbCr, September 20th, 2019].
- Jason Brownlee. A Gentle Introduction to the Bag-of-Words Model. October 9th, 2017 in Deep Learning for Natural Language Processing [https://machinelearningmastery.com/gentle-introduction-bag-words-model/, October 11th, 2019].
- Stephen O'Hara and Bruce A. Draper. Introduction to the Bag Of Features Paradigm for Image Classification and Retrieval. January 17th, 2011. [https://arxiv.org/pdf/1101.3354.pdf, October 11th, 2019].
- Cordelia Schmid. Bag-of-features for image classification. INRIA. [https://www.di.ens.fr/willow/events/cvml2011/materials/CVML2011_Cordelia_bof.pdf, October 13th, 2019].
- Bethea Davida. *Bag of Visual Words in a Nutshell: The art of choosing important features.* [https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb, October 13th, 2019].
- Robert Gove's Blocks. *Using the elbow method to determine the optimal number of clusters for k-means clustering.* December 26th, 2017. [https://bl.ocks.org/rpgove/0060ff3b656618e9136b, November 5th, 2019].
- Datanovia. Cluster Validation Essentials. *Determining The Optimal Number Of Clusters: 3 Must Know Methods.* [https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/, November 5th, 2019].
- AlindGupta. GeeksforGeeks. A computer science portal for geeks. *Elbow Method for optimal value of k in KMeans.* [https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/, November 5th, 2019].